UNIVERSIDAD DE EXTREMADURA

Escuela Politécnica

Ingeniero en Informática

Proyecto Fin de Carrera

Desarrollo de un sistema de comunicación para WSAN de bajo consumo

Enrique Carlos Mesías Llamazares

Septiembre, 2017

UNIVERSIDAD DE EXTREMADURA

Escuela Politécnica

Ingeniero en Informática

Proyecto Fin de Carrera

Desarrollo de un sistema de comunicación para WSAN de bajo consumo

Autor: Enrique Carlos Mesías Llamazares

Fdo.:

Director: Marino Linaje Trigueros

Fdo.:

Tribunal Calificador

Presidente:

Fdo.:

Secretario:

Fdo.:

Vocal:

Fdo.:

CALIFICACIÓN:

FECHA:

Una serie de seres animados,

objetos inanimados

y entes abstractos

han provocado efectos secundarios.

ÍNDICE GENERAL DE CONTENIDOS

Contenido

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

1 RESUMEN Y OBJETIVOS

En algunas investigaciones, proyectos o experimentos sobre entornos inteligentes se necesita realizar prototipos y/o pruebas. En este documento vamos a proponer un sistema para desarrollar prototipos y/o pruebas de redes de sensores y actuadores. Parte del presupuesto para realizar pruebas y prototipos, se ve afectado de forma lineal respecto al número de nodos de la red. Por ello utilizaremos algún SoC con capacidades inalámbricas de bajo presupuesto.

En primer lugar, estudiaremos las características generales de redes de sensores y actuadores inalámbricos *(WSAN)*. Observaremos que tienen características comunes con las infraestructuras de los sistemas ubicuos.

Utilizaremos la metodología de desarrollo en espiral para investigar y desarrollar un sistema para realizar prototipos de *WSAN*.

A medida que hemos profundizado en los ciclos de la espiral, nos hemos encontrado con diferentes riesgos a salvar. Las alternativas investigadas para abordar los problemas que surgían, han tomado un papel relevante en este documento.

Aunque ESP8226 tiene características deseables de una mota en WSAN, tiene un inconveniente principal. Este inconveniente está relacionado con el consumo energético debido a la utilización de WiFi.

Se investiga sobre formas de alargar la vida útil de un nodo, tomando relevancia la técnica de apagar y encender cuando sea necesario.

Por último, se pone a prueba el componente entregado para realizar prototipos de WSAN en ESP8226, bajo el entorno de desarrollo Arduino.

2 INTRODUCCIÓN

Las redes de sensores y actuadores, es una propuesta muy interesante para que el entorno sea inteligente y lo suficientemente autónomo para tomar la iniciativa de resolver ciertas incidencias. El desarrollo e investigación de las redes de sensores comenzó hace dos décadas.

La evolución de las redes de sensores ha sido impulsada por el desarrollo de redes inalámbricas y sistemas integrados cada vez más diminutos y con más capacidad computacional.

Las redes de sensores pasan a conocerse como redes de sensores inalámbricas *(WSN)*. Las redes de sensores inalámbricos se vuelven muy interesante para la extracción de datos de un sistema o entorno sin tener que desplazarse hasta el entorno.

Se han realizado numerosas investigaciones sobre redes de sensores inalámbricas, utilizando para las pruebas, dos alternativas.

Una alternativa es usar como herramienta un simulador de red.

La otra alternativa, es realizar el experimento con placas de desarrollo de propósito general, aptas para las redes de sensores inalámbricas. Estas placas de desarrollo han ido refinando su diseño y capacidades. Reduciendo su tamaño, integrando sistemas de alimentación, módulos de radiofrecuencia, añadiendo sensores integrados e interfaces de comunicaciones.

Actualmente están llegando diferentes SoC con módulos de radiofrecuencia para comunicarse inalámbricamente. Estos SoC están entrando en el mercado principalmente, por un precio reducido y la existencia de recursos en Internet para desarrollar proyectos. Proyectos asociados con automatización o desarrollo de objetos inteligentes y conectados a Internet.

Para realizar pruebas o experimentos de redes de sensores con actuadores (WSAN), puede ser interesante estudiar la viabilidad de utilizar estos SoC de bajo presupuesto.

3 ANTECEDENTES / ESTADO DEL ARTE

3.1 WSAN

Las redes de sensores y actuadores inalámbricos son capaces de monitorizar el entorno y tomar decisiones adecuadas. Los nodos interactúan para obtener datos y actuar en consecuencia.

En las redes de sensores y actuadores no es necesaria una entidad central para coordinar las funciones de los nodos. Existe la interacción sensor con actuador y actuador con actuador (1).

Esta necesidad de interacción requiere que WSAN cumpla los siguientes requisitos:

- **Heterogeneidad de nodos**
    - **Sensores**: Deben ser baratos y pequeños, esto implica menos capacidad de computación e incluso menos alcance de comunicación inalámbrica.
    - **Actuadores**: Son nodos con más capacidad de computo, con una mayor duración de la batería y más potencia de transmisión inalámbrica.
- **Tiempo real**
    - Necesidad de latencia baja para que los nodos actuadores puedan actuar dentro de un periodo de tiempo válido.
- **Despliegue**
    - Cientos de sensores por cada actuador, los nodos actuadores deben ser desplegado según la necesidad de la aplicación o área a cubrir de actuación.
- **Coordinación**
    - Es necesario algún mecanismo para una comunicación eficiente entre sensores y actuadores y entre actuadores y actuadores.

- **Movilidad**
    - Los nodos pueden tener movilidad, normalmente los nodos actuadores.

3.1.1 Arquitecturas físicas de WSAN

En WSAN hay dos propuestas, una arquitectura semi-automatizada y por otro lado una arquitectura automatizada. Las diferencias principales son la forma de comunicar la información del evento ocurrido y la coordinación de los nodos actuadores.

3.1.1.1 Arquitectura semi-automatizada

En la arquitectura semi-automatizada los nodos sensores informan a un punto central, de una forma similar a las redes exclusivamente de sensores. Posteriormente, el punto central comunica con los nodos actuadores para que realicen las acciones.

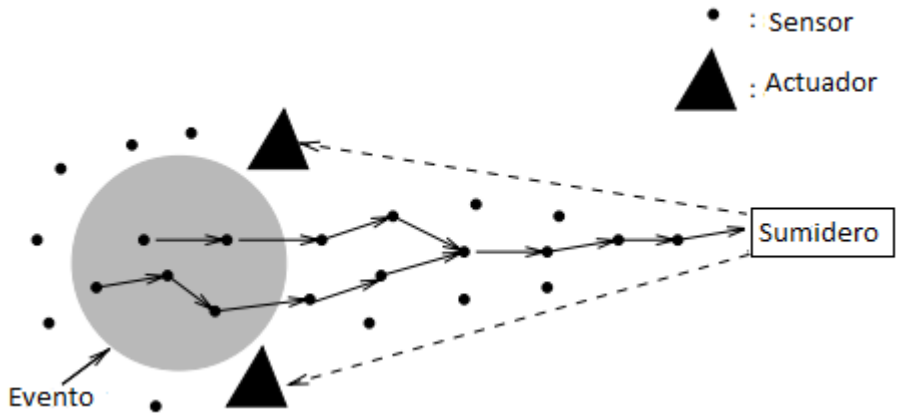

Ilustración 1.-Traducción de Arquitectura semi-automatizada (1)

3.1.1.2 Arquitectura automatizada

En la arquitectura automatizada los nodos sensores informan del evento al nodo o conjunto de nodos actuadores interesados en el evento. Posteriormente, los nodos actuadores realizan las acciones pertinentes y también pueden comunicarse para cooperar con otros nodos actuadores.

En la arquitectura automatizada se requieren algoritmos distribuido para que los nodos puedan comunicarse y coordinarse.

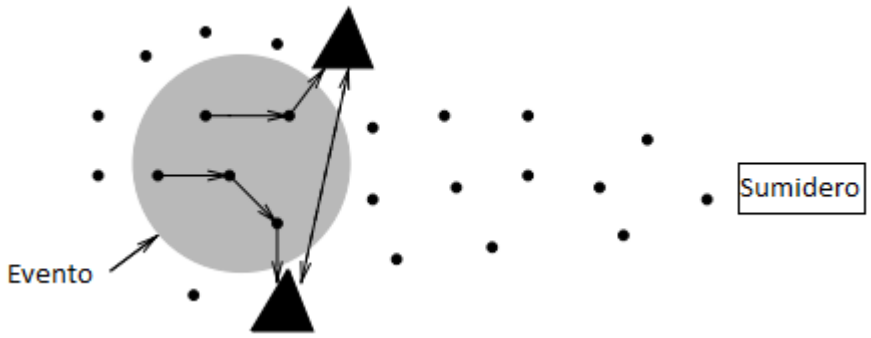

Ilustración 2..-Traducción de Arquitectura automatizada (1)

3.1.2 Ventajas de arquitectura WSAN automatizada frente a semi-automatizada

En WSAN automatizado la interacción entre nodos sensores y nodos actuadores ocurre de una forma local o dentro del área del evento, en cambio en la arquitectura WSAN semi-automatizada la información debe llegar hasta un punto central. Es por esto que la arquitectura WSAN automatizada tiene una serie de ventajas:

- **Baja latencia**: La información del evento realiza menos saltos de nodo a nodo.

- **Bajo consumo y aumento del tiempo de vida de la red:** Sólo interactúan los nodos dentro del alcance del evento. Reduciendo de esta forma, la actividad de otros nodos como enrutadores de la información del evento a nodos fuera del área del evento.

3.2 WSAN como infraestructura de Sistemas Ubicuos

Las redes de sensores y actuadores dispersas en el entorno se tornan claves para el desarrollo de las infraestructuras en los sistemas ubicuos (2). WSAN tiene una serie de características deseables para ser integrada en los sistemas ubicuos, porque permite que el entorno tome la iniciativa para realizar ciertas acciones.

La computación ubicua fue explicada por Mark Weiser en 1991 (3) como la capacidad de integrar en el entorno la computación, para permitir salir a los humanos del foco de atención de los computadores personales. De este modo, las personas se podían enfocar directamente en las tareas que estuvieran realizando. El símil propuesto es, al igual que el humano no necesita centrarse en el proceso de leer, sino que directamente es capaz de entender la información.

Weiser propone que el entorno tome la iniciativa para facilitar el día a día de las personas.

Weiser describe un día cualquiera en el personaje ficticio Sal, donde un despertador se comunica con la cafetera para que vaya preparando el café, una ventana despliega información sobre el estado del vecindario, un asistente le ayuda a llegar al trabajo. Sal también puede acceder a varios computadores como si fueran uno.

Weiser expuso algunos requisitos para que la computación salga del foco de atención, los procesadores deberían ir disminuyendo en tamaño y aumentando su capacidad de cómputo, además deben de comunicarse de forma inalámbrica.

Para que estas infraestructuras crezcan se necesitan dispositivos-centristas, dispositivos con capacidad de cómputo y de comunicación inalámbrica (2).

Las infraestructuras deben cumplir con ciertos requisitos.

Las infraestructuras deben diseñarse buscando portabilidad consiguiendo un bajo acoplamiento al entorno. Para que una infraestructura pueda ser desplegado en cualquier entorno debe cumplir con los siguientes requisitos (2):

- **Volatilidad**: Los componentes de la infraestructura pueden asociarse y desasociarse entre ellos. Pueden cambiar sus comunicaciones lógicas y el software residente. Deben soportar los fallos debido a que las comunicaciones no son permanentes y debido a otros cambios.
- **Adaptación**: Las infraestructuras pueden descubrir, y reconfigurar servicios para mantener entornos inteligentes.
- **Evolución**: El sistema debe ser capaz de incorporar elementos no previstos en tiempo de ejecución.
- **Abierta**: Las infraestructuras deben permitir ser explotada por otras.
- **Energéticamente independientes**.
- **Reducir la barrera de entrada**: Permite investigar y experimentar las posibilidades del desarrollo de entornos inteligentes.

- **Desarrollo rápido**: Se han propuesto placas hardware de prototipado rápido.
- **Hardware económico**: Debe tener un coste económico bajo, porque una infraestructura de un sistema ubicuo está compuesta por una gran multitud de dispositivos y hardware integrado en el entorno.

4 METODOLOGÍA

4.1 Espiral

Hemos elegido para llevar a cabo nuestro proyecto la metodología de desarrollo en espiral propuesta por Boehm. La metodología en espiral es interesante para proyectos de carácter experimental.

Esta metodología se caracteriza por dedicar una fase para analizar posibles riesgos y desarrollar prototipos para evaluar la viabilidad de salvar el riesgo localizado. Además, el desarrollo es iterativo hasta alcanzar el grado deseado, esto permite ampliar las posibilidades del proyecto.

En nuestro proyecto nos enfrentamos a una multitud de riesgos técnicos debido a la inexperiencia en desarrollos de entornos inteligentes y a la dimensión exploratoria del proyecto.

La metodología consta de cuatro etapas en cada ciclo de desarrollo.

- **Planificación**: Se especifican los requisitos y restricciones del sistema a desarrollar. La planificación puede ir variando después de cada ciclo, para ajustarse a la viabilidad del proyecto y al grado de madurez que se pretenda alcanzar.

- **Análisis de riesgos**: En esta fase hay que analizar los riesgos que pueden encontrarse en el desarrollo de la planificación. Para salvar los riesgos se proponen distintas soluciones, realizando si fuera necesario prototipos para probar la posibilidad de salvar el riesgo.

- **Desarrollo**: Se desarrolla el producto o software para lograr las especificaciones de la planificación. También se realizan pruebas del desarrollo realizado.

- **Evaluación del desarrollo**: Se plantea el grado de madurez adquirido hasta el momento y se decide seguir con otro ciclo más de la espiral o en caso de haber logrado el nivel deseado de madurez, se termina el desarrollo.

Ilustración 3.-Etapas de un ciclo de desarrollo en metodología en espiral

5 IMPLEMENTACIÓN Y DESARROLLO

5.1 Restricciones y objetivos iniciales

Crear un sistema para realizar prototipos de WSAN como infraestructuras de sistemas ubicuos. La restricción principal es un coste económico bajo de las placas de desarrollo para superar la barrera de entrada que supone el coste de múltiples nodos.

5.2 Riesgo, Placa de desarrollo inadecuada

Una elección importante para el éxito de nuestros objetivos es elegir la placa de desarrollo adecuada. Para salvar el riesgo de elegir un hardware inadecuado, estudiaremos como debe ser el hardware en WSN (las redes de sensores), ya que sobre WSAN no existen tantos documentos, ni investigaciones sobre hardware.

5.3 Investigando el hardware en WSN para proponer una placa de desarrollo adecuada

5.3.1 WSN

Las redes de sensores con capacidad de comunicaciones inalámbricas han surgido con el propósito principal de adquirir y transmitir datos de un entorno hasta un destino. En los últimos años han sido estudiadas por diferentes investigadores, y son de utilidad en diversos dominios.

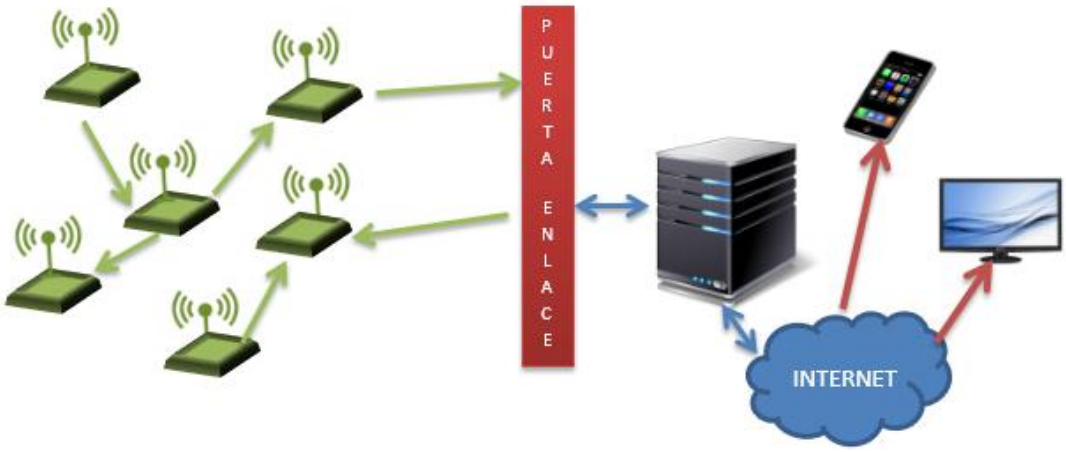

Ilustración 4.-Red de sensores inalámbricos

Las aplicaciones de WSN, al igual que algunos conocimientos y ciencias, se han explotado primeramente para uso militar (4). A medida que se ha reducido el coste de los sensores, microcontroladores y las comunicaciones, se ha extendido su aplicación para ayudar al desarrollo de investigaciones científicas, industrias, y posteriormente dispuesto al servicio de la sociedad.

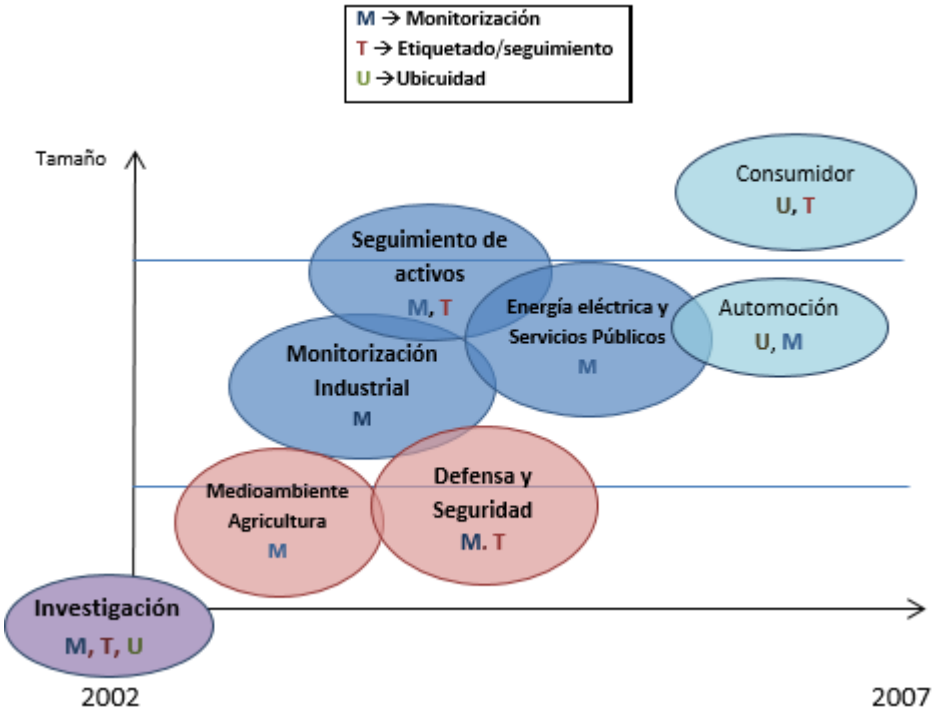

Ilustración 5.-Adopción de WSN (5)

5.3.2 Motas

Las motas son nodos que sensorizan el entorno con recursos de computación limitados, esto se debe a que se prioriza el tamaño y el bajo consumo. Disponen de un sistema de comunicación inalámbrico para volcar la información obtenida por el sensor a la red. Una mota en WSN está compuesta por cinco partes.

Ilustración 6.-Arquitectura hardware de un nodo WSN

5.3.3 Procesador

El procesador en los nodos suele ser de la escala de MHz, normalmente son microcontroladores que necesitan poca energía para su funcionamiento, ya que se busca un bajo consumo.

La función principal del microcontrolador es procesar la información adquirida por los sensores y la recibida por otros nodos, también permite gestionar las operaciones de los diferentes periféricos, así como la memoria y el chip de comunicación inalámbrica.

5.3.4 Memoria

La memoria es la encargada de almacenar los datos adquiridos por el sensor, además del espacio requerido por el proceso del nodo. Existen varios tipos de memorias (6) con diferentes características y con propósitos diferentes.

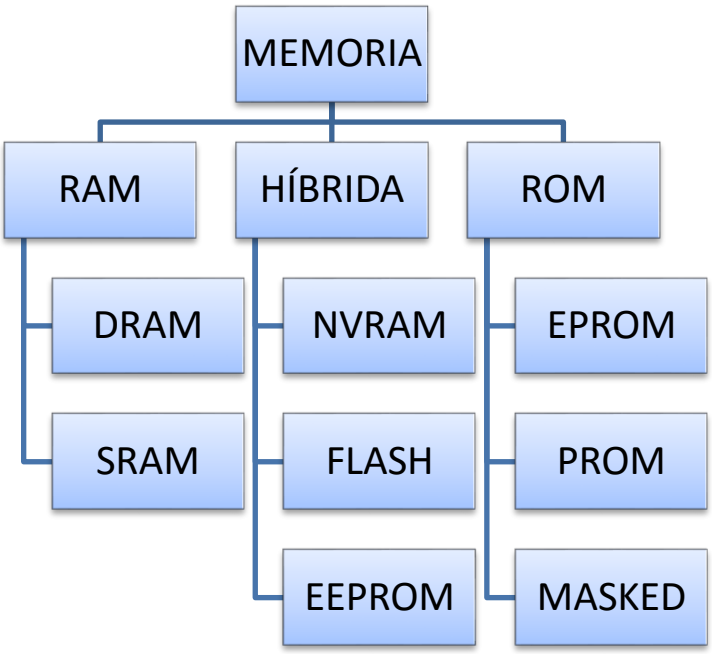

Ilustración 7.-Memorias para sistemas embebidos (6)

- **Tipos de RAM**: La RAM son memorias volátiles.
    - **SRAM**:RAM estática, necesita estar alimentada de forma continua, sino los datos se pierden.
    - **DRAM**: RAM dinámica, necesita ser refrescada mediante un controlador, puede permanecer hasta 4 milisegundos sin alimentación eléctrica. La DRAM tiene un precio mayor y menor velocidad de acceso que la SRAM.
- **Tipos de ROM**: La ROM es memoria no volátil.
    - **MASKED**: Es un tipo de ROM donde los datos e instrucciones son cableados físicamente, se diseña antes de producir los chips.
    - **PROM**: Es una ROM programable, puede ser programada una única vez.
    - **EPROM**: Es una ROM programable, esta permite ser borrada y ser reprogramada de nuevo.

- **Híbridos**: Estas memorias adquieren las capacidades tanto de una RAM como de una ROM, es decir permiten escrituras en tiempo de ejecución como una RAM y permite escribir información sin que esta se pierda al ser desconectada de la corriente eléctrica.
    - **EEPROM**: Son similares a las EPROM con la salvedad de que el borrado se realiza de forma eléctrica, sus lecturas son más lentas que en una memoria RAM. Esto implica que no se suele utilizar como memoria principal del sistema.
    - **FLASH**: Actualmente son memorias baratas, rápidas para lectura y con una buena densidad (capacidad de almacenamiento/superficie).
    - **NVRAM**: Es una SRAM con una batería que retiene la información, se suele utilizar para una pequeña cantidad de información crítica para el sistema.

5.3.5 Sensor

Para obtener la información del entorno la mayoría de los sensores se basan en la utilización de transductores (7). Un transductor es un dispositivo que, utilizando algún principio físico adquiere la energía eléctrica de forma analógica. Posteriormente la convierte en una señal digital. Algunos ejemplos de sensores están resumidos en la siguiente tabla.

Mediciones para redes de sensores inalámbricos	
	Medidas
Propiedades físicas	Presión
	Temperatura
	Humedad
	Flujo en corriente
Propiedades de movimiento	Posición
	Velocidad
	Velocidad angular
	Aceleración
Propiedad de contactos	Tensión
	Fuerza
	Torsión
	Desliz
	Vibración
Presencia	Táctil / contactos
	Proximidad
	Distancia / rango
	Movimiento
Bioquímica	Agentes bioquímicos
Identificación	Características personales
	Identificación de la persona

Tabla 1.- Sensores y medidas (7)

5.3.6 Batería

Es la encargada de alimentar los componentes del nodo (8), la gran parte del consumo energético se debe a la transmisión de información inalámbrica. Existen diferentes tipos de baterías (5):

- **Baterías alcalinas:** Son de uso común, existe una gran disponibilidad en el mercado y son suficientes para sistemas que no requieran un alto nivel de corriente. Su capacidad de suministrar energía disminuye cuando existen temperaturas bajas en el ambiente por debajo de los 10ºC.

- **Baterías de litio:** Son desechables, tienen un buen desempeño a temperaturas bajas y no es afectada por un aumento de la intensidad de la corriente. Pueden utilizarse al aire libre.

- **Baterías de hidruro de níquel-metal:** Son recargables y el tiempo de recarga va disminuyendo de forma ínfima, como así también lo hace su capacidad. Se ven afectadas a temperaturas de 40º bajo cero.

5.3.7 Comunicación inalámbrica

Los nodos integran un transceptor para enviar y recibir información de forma inalámbrica dentro de su rango de alcance.

Estos transceptores utilizan la banda ISM, esta banda es de utilización no comercial para áreas industriales, científica y médica. Su utilización está abierta a todo el mundo respetando los niveles máximos de potencia transmitida.

Los estados más comunes de los transceptores son emitir, recibir, dormir e inactividad. Normalmente el modo inactivo de los transceptores suele consumir igual que el modo recibir, por ello es recomendable apagar el transceptor cuando no se espera recibir información (8).

El estándar actual de comunicación que más se ajusta a las necesidades de las redes de sensores es IEEE 802.15.4 (9) se caracteriza por un bajo consumo de potencia a cambio de una baja tasa de transferencia de datos.

El IEEE 802.15.4 está dentro de en las redes inalámbricas de área personal.

IEEE 802.15.4 para reducir el consumo energético los nodos de la red despiertan de forma sincronizada para intercambiar la información y tras ello vuelven a dormirse.

IEEE 802.15.4 define las dos primeras capas del modelo OSI (Interconexión de sistemas abiertos), es decir, la capa física y la capa de enlace.

5.3.7.1 ZigBee

ZigBee es una implementación de IEEE 802.15.4 que añade la capa de red y de aplicación.

En ZigBee existen tres tipos de dispositivos (10):

- **Coordinador:** Gestiona la red y controla las rutas, existe solamente un coordinador por red.

- **Enrutador:** Interconecta diferentes nodos, es utilizado para ampliar la cobertura de la red en topologías arbóreas o mallas.
- **Dispositivo final:** Se conecta al enrutador o al coordinador, consume poca energía porque está activo un corto periodo de tiempo para transmitir información.

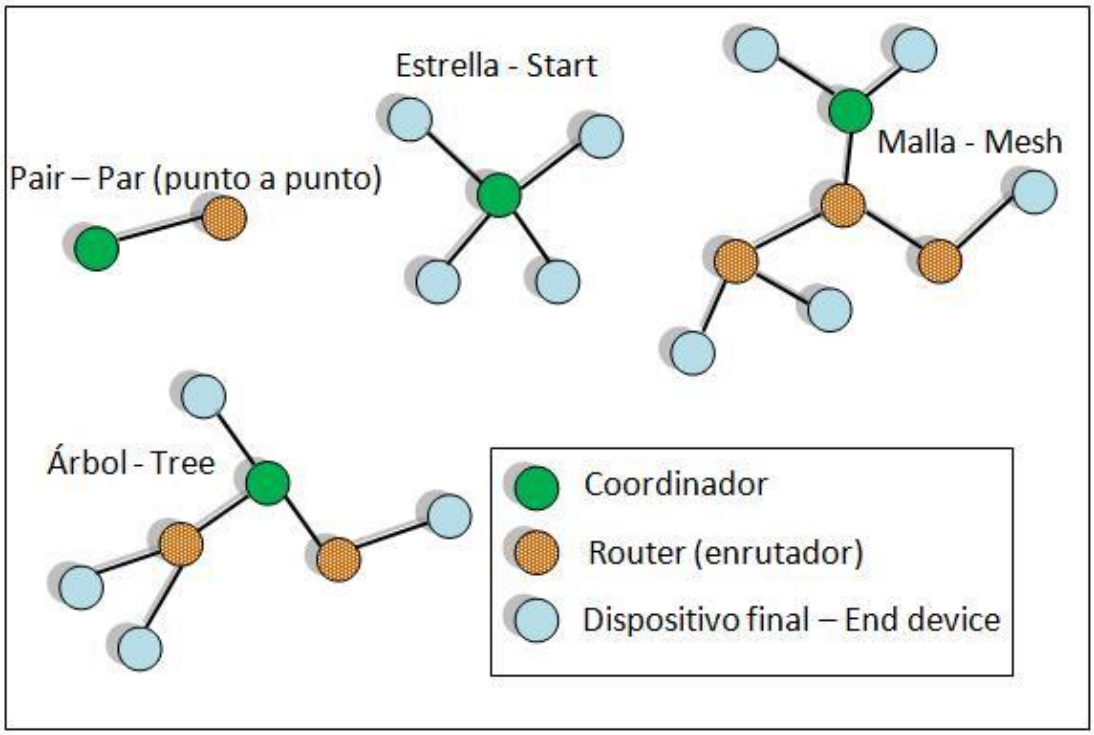

Ilustración 8.-Topologías en ZigBee (11)

5.4 Desarrollo, eligiendo la placa de desarrollo adecuada

Buscaremos en el mercado un SoC (System on Chip) que se ajuste a nuestras restricciones iniciales para alcanzar los objetivos propuestos. Para elegir el SoC más adecuado, debemos valorarlo a través de las principales características, de un modo similar al análisis comparativo que realizaron sobre motas para redes de sensores (12).

5.4.1 Características evaluables en motas WSN

5.4.2 Precio

El precio es importante para lograr una barrera de entrada baja, además de que en las redes de sensores y actuadores existe un alto número de nodos. El coste de la red es directamente proporcional al número de nodos.

La mayoría de las motas desarrolladas específicamente para WSN tienen un precio alrededor de cien euros. Vamos a proponer placas de desarrollo que son válidas para desarrollar redes de sensores y actuadores. Aunque estas placas no disponen de un sistema de alimentación integrado, se puede añadir posteriormente.

En la siguiente tabla mostramos diferentes placas de desarrollo con sus características principales:

Características	Arduino nano + Xbee+shield	Pi Zero W	ESP8226 NodeMCU
Precio*	5+25+5=35 euros	11 euros	3 euros
Tamaño(Superficie)	60mmx62mm(shield)	65mm × 30mm	51mm × 23mm
Peso	7gr+5.7gr	9gr	11gr (13)
Comunicación Inalámbrica	"ZigBee"	802.11n + Bluetooth 4.1 y Bluetooth LE	802.11n
Consumo de la placa sin conectividad	5V*35mA (14)	5V*80mA (15)	5V*20mA
Interfaces	14 GPIO 8 analógicos I/O 1 UART 6 PWM I2C/SPI (16)	40 GPIO	11 GPIO 1 analógico I/O 1 UART 9 PWM I2C/SPI (16)
Comunidad	Diseñada con propósitos educativos	Comunidad Linux + propósitos educativos	Salida reciente, comunidad en crecimiento

Tabla 2..-Características de algunos SoCs conocidos en el mercado

**Precios orientativos consultados en tiendas online internacionales a mediados de agosto de 2017*

5.4.3 Tamaño y peso

En las redes de sensores inalámbricos se busca que el hardware del nodo sea lo más diminuto posible, para poder ser desplegado en cualquier entorno tratando de ser imperceptible.

Un nodo con poco peso es más fácil de manipular y de soportar sobre otros lugares.

5.4.4 Comunicación inalámbrica

La tecnología utilizada para transmitir y recibir información define principalmente el alcance, consumo energético, ancho de banda, el número máximo de nodos e incluso la disposición o topología de la red de sensores.

Veamos una tabla comparativa entre Bluetooth, ZigBee, WiFi con información extraída de (17).

Tecnologías inalámbricas			
Conocidas como	**Bluetooth**	**ZigBee**	**WiFi**
Éstandar	IEEE 802.15.1	IEEE 802.15.4	IEEE 802.11
Área	Red de área personal inalámbrica (WPAN)	Baja tasa WPAN (LR-WPAN)	Red de área local inalámbrica (WLAN)
Alcance	10m	10-100m	100m
Topología	Piconet Scatternet	Malla Árbol Estrella	BSS ESS
Número de nodos	8	65000	2007
Ancho de banda	1Mbit/s	250 Kbit/s	54 Mbit/s
Corriente necesaria transmitiendo	57mA	25mA	220mA

Tabla 3.-Comparación Bluetooth, ZigBee, WiFi

Se aprecia en la Tabla 3 que ZigBee tiene características más adecuadas para redes de sensores, como se indicó anteriormente.

5.4.5 Consumo de la placa sin conectividad

Es importante conocer el consumo energético debido al tratamiento y procesamiento de los datos, normalmente para las tareas internas del nodo.

5.4.6 Interfaces

Permite y facilita la extensión de las funcionalidades del nodo, añadiendo sensores y actuadores.

5.4.7 Comunidad

Una comunidad compuesta por una gran cantidad de personas implica más cantidad de recursos, experimentos, proyectos y empresas interesadas en desarrollar nuevos componentes y mejoras para la plataforma. Todo ello conlleva a un desarrollo más eficaz y que la plataforma evolucione más rápido.

5.4.8 Evaluación de los SoC propuestos

Tras revisar las características interesantes en el hardware para WSN, hemos evaluado de forma cualitativa mediante colores, que placa de desarrollo cumple mejor X característica con respecto a la misma característica X en las otras placas de desarrollo evaluadas.

Características	Arduino nano + Xbee+shield	Pi Zero W	ESP8226 NodeMCU
Precio	5+25+5=35 euros	11 euros	3 euros
Tamaño(Superficie)	60mmx62mm(shield)	65mm × 30mm	51mm × 23mm
Peso	7gr+5.7gr	9gr	11gr (13)
Comunicación Inalámbrica	"ZigBee"	802.11n + Bluetooth 4.1 y Bluetooth LE	802.11n
Consumo de la placa sin conectividad	5V*35mA (14)	5V*80mA (15)	5V*20mA
Interfaces	14 GPIO 8 analógicos I/O 1 UART 6 PWM I2C/SPI (16)	40 GPIO	16 GPIO 1 analógico I/O 1 UART 9 PWM I2C/SPI (16)
Comunidad	Diseñada con propósitos educativos	Comunidad Linux + propósitos educativos	Salida reciente, comunidad en crecimiento
Puntuación total	**3*1+2*2+2*3** = 13	**2*1+3*2+2*2** = 14	**2*1+2*2+3*3 = 15**

Tabla 4.-Evaluación de los SoCs conocidos para WSN

5.5 Evaluación del primer ciclo de la espiral

Hemos elegido ESP8226 por tener mejores características que las otras dos alternativas propuestas, la ventaja principal es un bajo precio, un consumo de procesamiento bajo y algunas interfaces para agregar sensores, actuadores, etc.

El punto más negativo es el consumo energético debido a la comunicación inalámbrica de WiFi, por otro lado, la comunidad no es tan grande como las otras plataformas.

Realizaremos otro ciclo más en la espiral, para tratar de ver si somos capaces de solventar en cierto modo estos aspectos negativos.

5.6 Restricciones y objetivos, segundo ciclo

Nuestro objetivo sigue siendo desarrollar un sistema para realizar prototipos de infraestructuras WSAN para sistemas ubicuos, pero debemos de ser capaces de solventar de algún modo el consumo energético de la comunicación inalámbrica impuesta por WiFi.

5.7 Riesgo, ESP8226 entorno desconocido

Es interesante estudiar más a fondo las capacidades del ESP8226, para buscar si existe alguna forma de reducir el gasto energético del radio WiFi.

5.8 Desarrollo, conociendo el ESP8226

El ESP8226 (18) es un SoC con WiFi(b/g/n) integrado con un uso eficiente de la energía utilizada en un diseño compacto y pensado para el internet de las cosas.

Integra un microcontrolador de 32 bits Tensilica L106 y un conjunto de instrucciones RISC de 16 bits. La velocidad de la CPU es de 80MHz pudiendo alcanzar un máximo de 160MHz acompañado de una SRAM compuesta de 64 KB para almacenar instrucciones y 96 KB para datos data.

Integra una memoria flash mediante SPI entre 512KB y 16MB como máximo.

El radio WiFi puede estar en modo punto de acceso, en modo estación o en ambos bajo el mismo canal. Soporta seguridad WPA/WPA2 además de encriptación WEP. Trabaja con protocolos IPv4, TCP/UDP implementados con LWIP (19) que es una versión ligera y configurable de la pila TCP/IP para sistemas embebidos.

El rango de operación según la temperatura es de -40°C ~ 125°C. El voltaje de operación está entre 2.5 y 3.6 voltios.

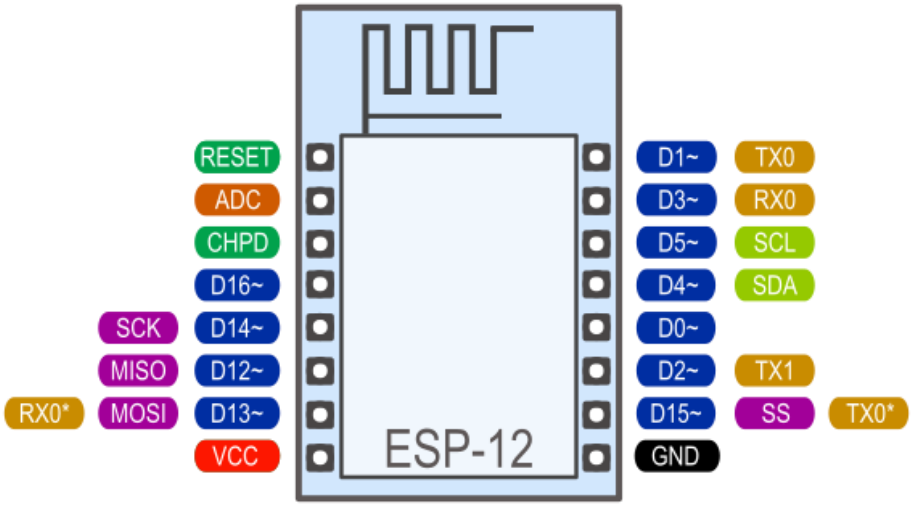

Ilustración 9.-Esquema de un ESP8226-12 (20)

5.8.1 Modos y consumos de energía en ESP8226

Espressif ha diseñado varios modos para el ESP8226, estos modos de bajo consumo van desactivando módulos del sistema para ahorrar energía (21).

En la Tabla 5 vemos como existen diferentes potencias y tipos de protocolos WiFi, además de los modos de ahorro con sus respectivos consumos.

Consumo de energía		
Parámetros	**Media de consumo**	**Unidad**
Tx 802.11b, P= +17 dBm	170	mA
Tx 802.11g, P= +15 dBm	140	mA
Tx 802.11n, P= +13 dBm	120	mA
Rx 802.11b, -80 dBm	50	mA
Rx 802.11g, -70 dBm	56	mA
Rx 802.11n, -65 dBm	56	mA
Modem-sleep	15	mA
Light-sleep	0,9	mA
Deep-sleep	20	µA
Power off	0,5	µA

Tabla 5.-Modos y consumos energéticos de ESP8226 (18)

En modo ‘modem sleep’ y en el modo ‘light sleep’ el sistema decide cuando va a descansar durante unos milisegundos.

En WiFi existe un mecanismo para ahorrar energía, conocido como DTIM. El router o punto de acceso antes de enviar información envía una notificación para indicar cuando deben escuchar los clientes para recibir datos. Estos modos cierran el circuito WiFi entre intervalos para consumir menos energía.

En el modo ‘deep sleep’ el nodo queda en stand by esperando a que se active una señal mediante un pin, de esta forma el usuario puede asegurarse de la cantidad de tiempo que descansará el nodo.

En la Tabla 6 observamos como el modo estación del wifi se desactiva y a medida que se quiere ahorrar más energía se van desactivando funcionalidades del sistema.

Item/Modos:	Modem Sleep	Light Sleep	Deep Sleep
WIFI	OFF/STATION	OFF/STATION	OFF
RELOJ SISTEMA	ON	OFF	OFF
RTC	ON	ON	ON
CPU	ON	Pendiente	OFF
CONSUMO	15mA	0,9 mA	20µA
DECIDE	Sistema	Sistema	Usuario

Tabla 6.-Modos de bajo consumo ESP8226

5.8.2 Plataformas de desarrollo software

En menos de cinco años se han desarrollado diferentes entornos de desarrollo para ESP8226, esto implica que existe una comunidad de desarrolladores que va en aumento. A continuación, exponemos algunos entornos de desarrollo:

- **Librerías Espressif:** Son las entregadas por el fabricante para programar el ESP8226, están desarrolladas en C. Estas librerías soportan hasta tres colas de prioridad diferentes para las tareas, aunque estas tareas no son *preemptivas*, es decir no pueden ser interrumpidas. Son las más rápidas puesto que trabajan a bajo nivel, pero el desarrollo es más complejo.

- **Portado a RTOS:** Estas librerías han sido portadas para poder usarse en FreeRTOS, que es un micro kernel para microcontroladores para soportar sistemas en tiempo real.

- **Intérprete Lua:** Curva de aprendizaje más fácil, pero necesita RAM para alojar el intérprete Lua.

- **Plataforma adaptada a Arduino:** Mediante las librerías Espressif en C, han desarrollado las APIS de Arduino para hacer que ESP8226 sea compatible con una gran variedad de librerías de Arduino. Existen una gran variedad de librerías en Arduino para gestionar otros periféricos, sensores, pantallas, etc. Además, se puede desarrollar mediante el IDE de Arduino y se puede utilizar las librerías bases de Espressif.

- **Comandos AT:** No es un entorno de desarrollo, pero es un sistema de comunicación por UART para ser integrado por otros controladores. El consumo de memoria es ínfimo.

Plataformas de desarrollo	Curva de aprendizaje	Librerías añadidas	Consumo de memoria	Comunidad
Espressif	Alta	Baja	Bajo	Mediana
RTOS	Muy Alta	Media	Medio	Pequeña
LUA	Baja	Alta	Alto	Grande
ARDUINO	Media	Muy Alta	Medio	Muy grande
AT	Media	Nula	Poco	Pequeña

Tabla 7.-Comparativa IDE para ESP8226

5.9 Evaluación del segundo ciclo de la espiral

ESP8226 tiene buenas prestaciones de cómputo y memoria en relación con su precio. Por otra parte, dispone de diferentes plataformas de desarrollo, entre ellas la más destacable es la plataforma de desarrollo en Arduino.

La plataforma de Arduino solventa uno de los inconvenientes expuestos en la Tabla 4 , en concreto el relacionado con la comunidad existente para ESP8226. Se pueden aprovechar algunas librerías que han sido desarrolladas para Arduino para nuestro ESP8226. Además de disponer de un IDE (entorno de desarrollo) afable.

Analizando los modos de operación del ESP8226 podemos apreciar que; en condiciones donde esté constantemente comunicándose con otros ESP8226, puede tener un gran impacto en el consumo energético.

Vamos a realizar otro ciclo en la espiral para encontrar soluciones para que el nodo pueda ser independiente energéticamente durante un tiempo.

5.10 Restricciones y objetivos, tercer ciclo de la espiral

Al conocer más de cerca al ESP8226 vemos las limitaciones y las bondades de este SoC. Para reducir el consumo energético debemos tener en cuenta el modo dormir del ESP8226 y realizar las comunicaciones entre los nodos sólo cuando sea necesario. También hay que tener en cuenta la limitación de memoria, flash y RAM para gestionar la cantidad de mensajes que puede soportar el nodo.

En cambio, en el aspecto de integración de la plataforma con otros periféricos y librerías, se está realizando un gran esfuerzo al portar Arduino para ESP8226.

Vamos a desarrollar un componente software utilizando el entorno de Arduino IDE con los siguientes objetivos:

- Portable a otras plataformas.
- Despliegue sencillo, sin necesidad de nodos centrales y que los nodos puedan entrar, salir y caerse de la red sin que la red deje funcionar.
- Comunicación mediante saltos.
- Alargar la vida útil de la batería del nodo.
- Sin necesidad de un reloj global.
- Abierto y flexible, para que pueda ser utilizado por otros entornos.

5.11 Riesgo, alargar la vida útil de la batería

Antes de empezar a desarrollar un componente para que los nodos ESP8226 puedan comunicarse teniendo en cuenta los requisitos previos, vamos a investigar sobre las formas de alargar la vida útil de un nodo en WSN.

5.12 Investigando, alargar vida útil de los nodos en WSN

Para tratar de alargar la vida de la red de sensores e incluso hacer que un sensor sea autosuficiente energéticamente, existen varias propuestas que pueden ser aplicadas (9). Principalmente son recolectar energía del entorno del nodo, la forma de encaminar los mensajes por la red y planificar cuando el nodo debe apagarse y encenderse cuando sea necesario.

5.12.1 Cosechar energía

Cosechar energía (22) o más conocido como *Harvesting energy*, se trata de alimentar un circuito mediante la energía capturada por fuentes en el entorno. La energía capturada debe ser transformada en energía eléctrica para alimentar el circuito o dispositivo.

Normalmente la energía eléctrica se almacena en una batería recargable para cuando la energía en el entorno disminuya, el dispositivo pueda seguir alimentándose de las reservas de energía eléctrica almacenada.

En un sistema de cosechar energía hay tres componentes claves:

- **Transductor o cosechador**: El transductor convierte la energía provista por la fuente, en energía eléctrica.
- **Almacenamiento de energía**: Una batería o un buen capacitor.
- **El controlador de energía**: Se encarga de entregar la energía eléctrica de la forma adecuada al sistema.

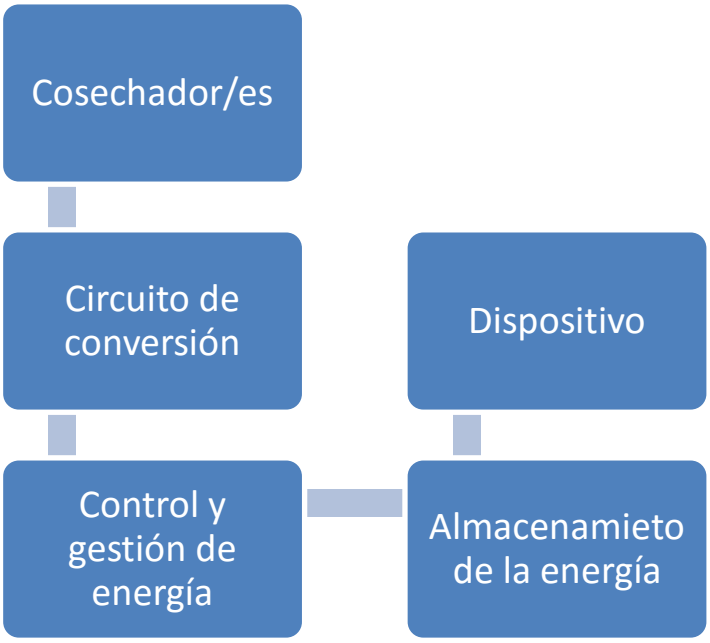

Ilustración 10.-Componentes en un nodo con cosechador de energía (22)

Las fuentes de energía más comunes son las siguientes:

- **Cinética:** Los transductores piezoeléctricos producen energía cuando tienen vibraciones, movimientos, ruido de motores, etc. Este transductor convierte la energía cinética en una tensión de salida analógica y continua, que es rectificada, regulada y almacenada en un condensador.
- **Térmica:** La tensión es producida por la diferencia de temperatura entre la unión de dos conductores o semiconductores diferentes. El sistema para recolectar la energía consiste en una matriz de termopares conectados en serie a una fuente de calor. Las fuentes de calor pueden ser calentadores de agua, un motor, la parte posterior de un panel de placa solar, el espacio entre un componente emisor de calor y el disipador, etc.

- **Solar:** Mediante células fotovoltaicas convierten la luz en energía eléctrica, normalmente se utilizan acompañadas de una batería ya que la intensidad de la luz puede variar debido a condiciones climatológicas.

- **Radiofrecuencia:** Una antena receptora convierte las señales de radiofrecuencia en una tensión continua.

5.12.2 Esquemas de despertados

Una propuesta para reducir el consumo de los nodos cuando tienen que transmitir información en la red, es dormir el nodo cuando no tiene que enviar información.

Existen una gran variedad de esquemas de despertados, normalmente con el inconveniente de introducir retardos en la red. Estos retardos se deben a que los nodos comunican cada cierto tiempo.

Anastasi divide los esquemas en dos aproximaciones (23), gestión de la energía y control de la topología. En el control de la topología, el objetivo es seleccionar el conjunto de nodos que serán despertados, mientras que, en la aproximación de la gestión de energía, el objetivo es seleccionar en que intervalos los nodos deben despertarse dentro de un rango de un tiempo continuo.

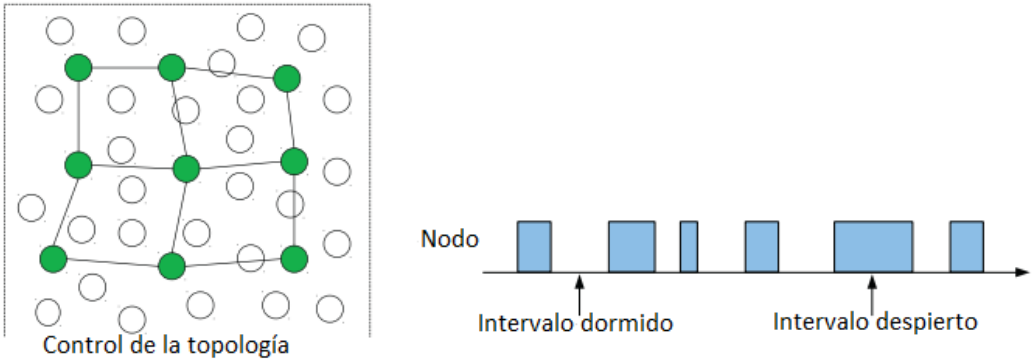

Ilustración 11.-Control de topología y gestión de energía del nodo (24)

El aproximamiento de gestión de la energía es comúnmente utilizado en la capa MAC, ya que también intentan no generar ruidos en las bandas de frecuencias debido al intento de varios nodos intentando transmitir al medio.

Existe una clasificación para los diferentes esquemas para la planificación de despertados, podemos ver una taxonomía (24) en la Ilustración 12.

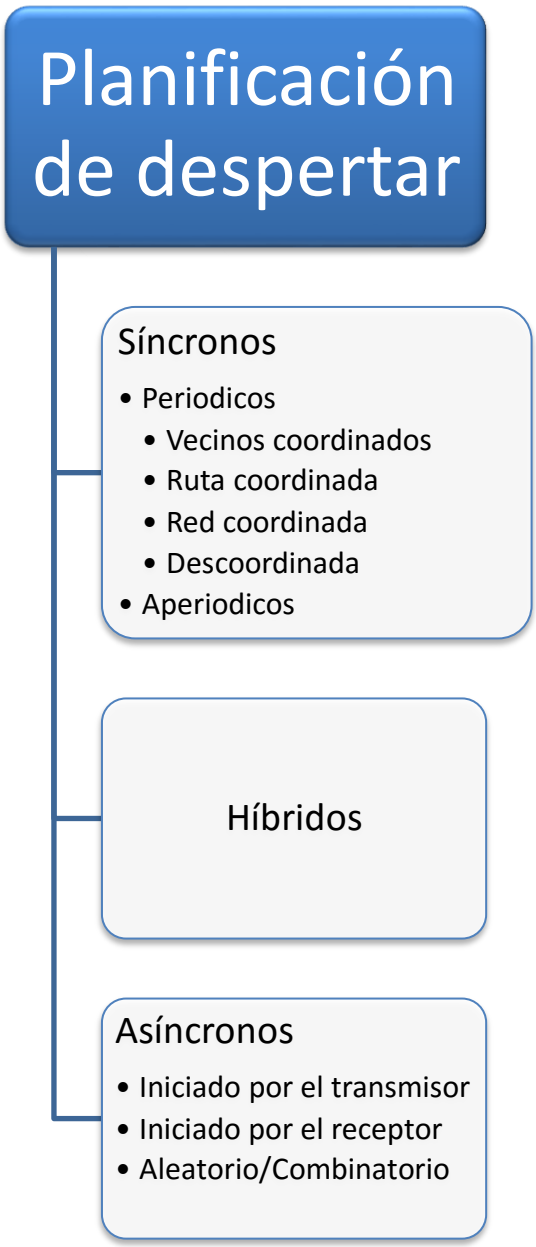

Ilustración 12.-Taxonomía Planificacion de despertar

Los esquemas síncronos periódicos están basados en ciclos y deben determinar en que momento del ciclo deben despertarse para transmitir o recibir paquetes. Esta decisión es más compleja en función del número de nodos involucrados, la coordinación de los nodos puede darse a tres niveles:

1. **Vecinos coordinados:** En este esquema el nodo decide su planificación en base a las planificaciones de los nodos vecinos.

2. **Camino coordinado:** Debido a las latencias introducidos por los esquemas de despertado, se propone que los nodos de una ruta se coordinen para transmitir la información.

3. **Red coordinada:** Es el más complejo de coordinar, la coordinación puede establecerse mediante un nodo dedicado a ello o mediante planificaciones locales.

También existen esquemas síncronos periódicos descoordinados, se basan en la información del propio nodo, como el aumento y la reducción de su propia cola de mensajes de envío y recepción.

En los esquemas periódicos asíncronos los nodos conocen la planificación de sus nodos vecinos para recibir o enviar mensajes.

Los esquemas asíncronos no necesitan almacenar la información de los nodos vecinos, simplemente esperan durante un tiempo para recibir mensajes o esperan un tiempo para encontrar un nodo al que enviar mensajes. En estos esquemas de despertado no es posible determinar el tiempo que va a estar dormido o encendido el nodo.

5.12.3 Protocolos de enrutamiento

Se han estudiado y desarrollado diversas topologías y diversos protocolos de enrutamiento para minimizar el impacto energético (25). Estos protocolos se pueden clasificar según diferentes criterios, pero los criterios más seguidos para su clasificación pueden verse en la Ilustración 13.

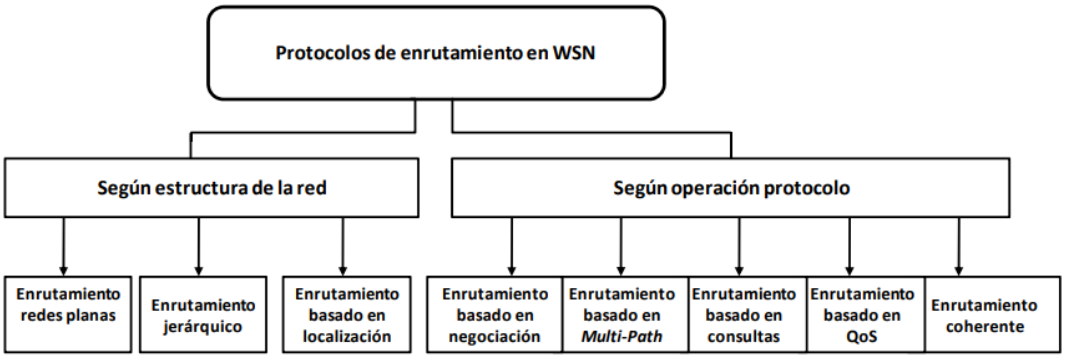

Ilustración 13.-Taxonomía de clasificación en Protocolos de enrutamiento WSN (25)

5.12.3.1 Estructura de la red

Los protocolos de enrutamiento se pueden clasificar según la disposición y las funciones de los nodos.

- **Estructura plana:** Los nodos en la red tienen las mismas funciones todos sensorizan y todos deben enrutar cuando sea necesario.
- **Estructura jerárquica:** Existen nodos que se encargan de encaminar los mensajes, coordinar nodos dentro de su clúster y otros son los nodos sensores. En las redes jerárquicas la caída de algún nodo coordinador puede afectar a la red entera.
- **Basado en la localización:** Se aprovecha el conocimiento de la geolocalización de los nodos en la red para enrutar los mensajes de forma eficiente.

5.12.3.2 Operación del protocolo

Los protocolos de enrutamiento WSN utilizan diferentes técnicas para encaminar los mensajes.

- **Negociación:** Se utilizan mensajes previos antes de enviar los mensajes de eventos a los nodos interesados, para evitar duplicaciones y redundancias de mensajes.
- **Múltiples rutas:** Utilizan diferentes rutas para alcanzar el destino.
- **Consultas:** Tratan a la red como una base de datos, propagan la consulta por la red y la desmenuzan hasta que alcanza los nodos que tienen las respuestas que serán devueltas.
- **QoS:** La red debe satisfacer ciertas métricas como latencia, energía y ancho de banda.
- **Coherente:** La información es enviada tras haber sido procesada por los nodos intermedios, que se encargan agregar y comprimir la información para obtener algoritmos más eficientes en términos energéticos.

5.12.3.3 Protocolos de enrutamiento en WSN

Veremos algunos protocolos que son descentralizado, de estructura plana ya que estos tienen la ventaja de que cualquier nodo puede ser desplegado sin importar su posición. Normalmente estos protocolos suelen estar basados en negociación o en consultas. Por otra parte, son más ineficientes que los jerárquicos, pero los jerárquicos no facilitan el despliegue sin importar la posición.

5.12.3.3.1 Flooding

Flooding es el protocolo más rudimentario simplemente un nodo envía una copia de un mensaje a todos sus nodos vecinos, sobrecarga la red de mensajes, esto aumenta la latencia.

Es el más sencillo de implementar y en caso de que no se descarten paquetes, debido al aumento de mensajes duplicados por la red, asegura que el paquete alcance los destinos interesados en él.

Existen dos problemas principales en flooding:

- **Implosión**: Debido a la propia naturaleza de enviar mensajes a todos los vecinos, un nodo que sea también vecino puede recibir mensajes duplicados.

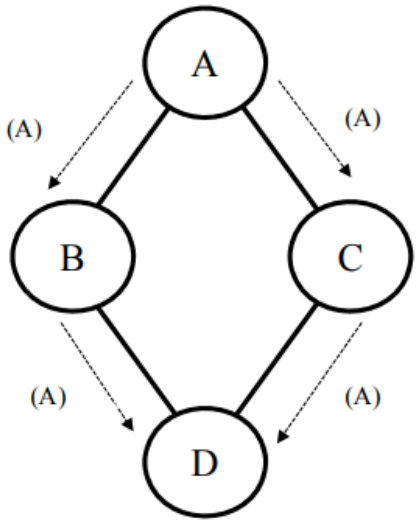

Ilustración 14.-Implosión en flooding (25)

- **Solapamiento**: Es una consecuencia de la implosión.

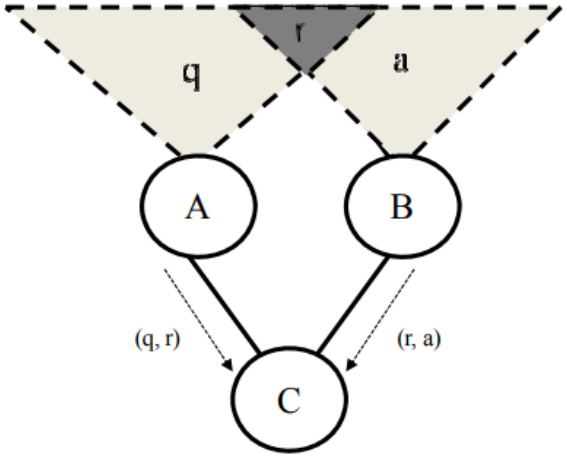

Ilustración 15.-Solapamiento en flooding (25)

Flooding empeora si los nodos en la red emiten de forma continuada muchos mensajes.

5.12.3.3.2 GOSSIPING

Se basa en los mismos principios que flooding, pero se selecciona un número de vecinos o existe algún mecanismo para controlar la saturación en la red.

5.12.3.3.3 SPIN

Este protocolo de enrutamiento está basado en negociación, el nodo que va a emitir un mensaje pregunta previamente que nodos vecinos están interesados en el mensaje. Recibe el interés del vecino y posteriormente envía el mensaje. De esta forma se evita recibir el mismo mensaje de forma duplicada.

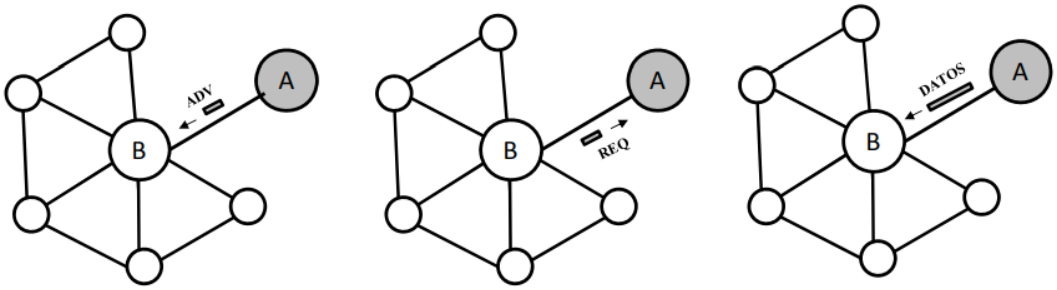

Ilustración 16.-Tres pasos para enviar un mensaje en SPIN (25)

El principal problema de SPIN es que puede haber nodos que no estén interesados en recibir el mensaje, siendo el nodo desinteresado parte de una ruta crítica para un nodo que si este interesado en ese mensaje, pero fuera del alcance del emisor.

5.12.3.3.4 Difusión directa

La difusión directa es un protocolo basado en consulta, el sumidero o la estación central envía un interés por conocer la información de un nodo. Este interés se propaga hasta alcanzar el nodo sensor. El nodo objetivo devuelve la información hacia el sumidero mediante saltos.

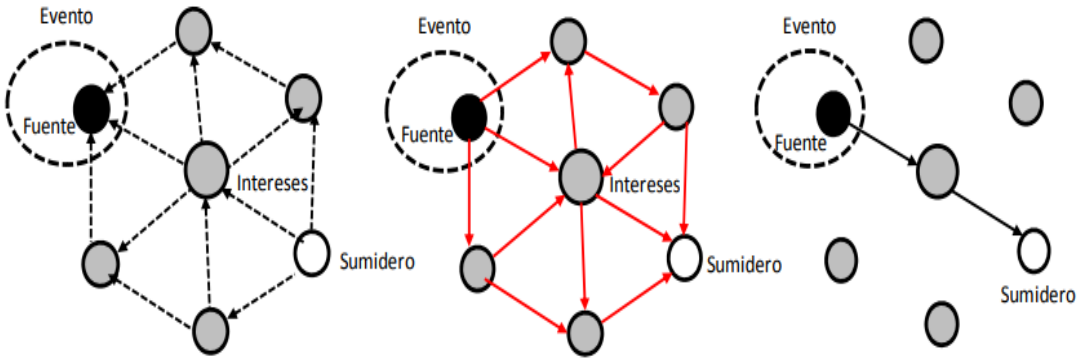

Ilustración 17.-Difusión directa (25)

La ventaja de difusión directa es que solo se transmiten mensajes bajo demanda de un punto, desde el sumidero. Pero si hubiera varios puntos interesados se comportaría de forma similar al flooding.

5.13 Desarrollo del sistema de bajo consumo para ESP8226

En el diseño de nuestro sistema tratamos de evitar el tiempo activo del radio de comunicación inalámbrica para reducir el gasto energético del nodo.

Utilizamos un esquema de despertado periódico asíncrono, esto permite que cada nodo tenga su propia duración de periodo. Es necesario conocer el periodo del nodo vecino para saber cuándo puede recibir mensajes.

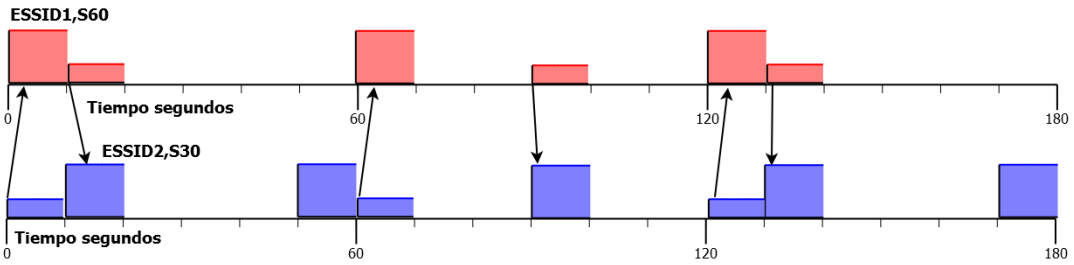

Ilustración 18.-Esquema de despertado periódico asíncrono

El SSID contiene de forma codificada el tiempo del periodo del nodo, en segundos. Para que no sea necesario esperar a recibir la información del periodo del nodo vecino.

5.13.1 Estados del nodo

Dentro del periodo de un nodo, un nodo puede realizar las siguientes funciones:

- **Anunciar y recibir mensajes:** En este estado el nodo está en modo punto de acceso esperando para recibir mensajes. Los mensajes se van almacenando en una cola de recepción.

  En el ESSID se expone en segundos el periodo del nodo, para que otros nodos vecinos conozcan el horario periódico del nodo.

  Es conveniente que la duración del estado recibir, sea estipulado por igual para todos los nodos, esto, determina el tiempo máximo para enviar mensajes a un nodo.

- **Escanear nodos vecinos:** En este estado el nodo está un tiempo escaneando el medio para encontrar nodos vecinos, cuando detecta algún nodo vecino guarda en una lista de vecinos el tiempo local en el que apareció y desgrana del ESSID vecino el tiempo de periodo.

  Por ejemplo, supongamos que aparece un vecino nuevo cuando estamos escaneando, el ESSID contiene al final un numero precedido por una. Suponga que es 'S30' esto indica que cada treinta segundos volverá a estar disponible para recibir mensajes. Es importante almacenar en que tiempo (local al nodo que escanea) fue visto.

  Con este par de información se puede calcular cuánto tiempo falta hasta el siguiente anuncio del nodo vecino, a través de la siguiente fórmula:

$$tiempo\ hasta\ el\ siguiente\ anuncio\ del\ nodo\ vecino = periodo\ del\ nodo\ vecino - ((tiempo\ local\ actual - tiempo\ local\ registrado\ en\ el\ que\ fue\ visto\ el\ nodo\ vecino)\%periodo\ del\ nodo\ vecino)$$

O de forma gráfica:

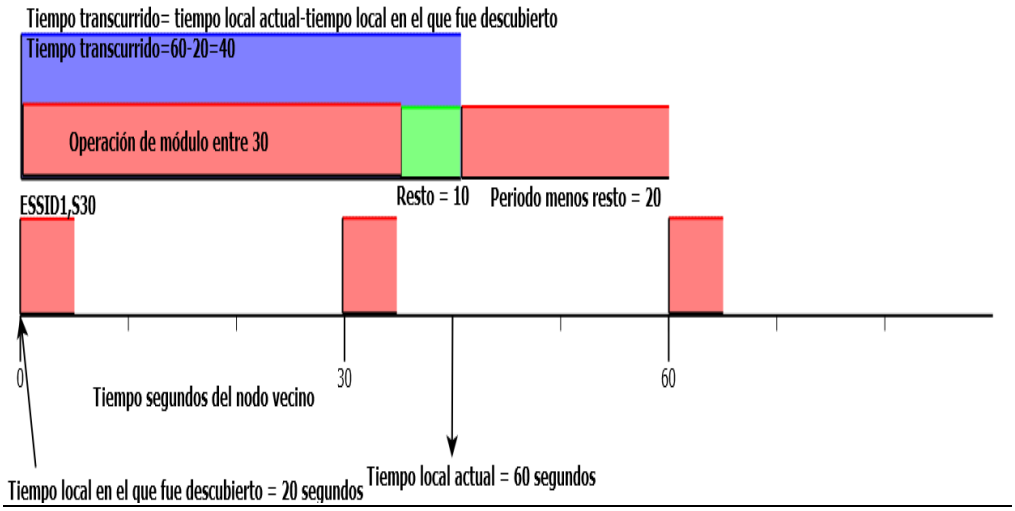

Ilustración 19.-Explicación gráfica del cálculo de los segundos hasta poder enviar al nodo vecino

Para asegurarse de que se han escaneado todos los vecinos, hemos establecido un máximo de tiempo para el periodo de cualquier nodo de la red. De esta forma escaneando durante ese máximo de tiempo encontraremos cualquier nodo vecino.

Buscar nodos vecinos es una operación que gasta mucha energía, por lo que hay que tratar de evitar su utilización. Por ello, solamente se utiliza al iniciar el nodo por primera vez y cuando no es capaz de enviar mensajes a un nodo vecino tras varios intentos. Bien porque se haya movido el propio nodo o el nodo vecino.

- **Enviar a un nodo vecino mensajes:** En cada periodo se intentará enviar mensajes a un nodo vecino. De esta forma dentro de un periodo como máximo habrá un tiempo destinado al envío. Si al realizar el envío

- **Realizar tareas de mantenimiento y de usuario**

    - **Revisar duplicados y mensajes que hayan estado en el nodo anteriormente**: Revisan los mensajes que hemos recibido, si los tenemos duplicado en alguna cola de mensajes del nodo lo eliminamos. También si han estado en el nodo anteriormente lo eliminamos. En el caso de que el mensaje de la cola de recibido no esté ni duplicado, ni haya estado en el nodo, le añadimos al mensaje el id propio del nodo y lo ponemos en la cola de mensajes listo para leer.

    - **Eliminar mensajes de la cola de envío:** Se eliminan aquellos mensajes que se hayan enviado a todos los nodos vecinos que existan en la lista de vecinos.

    - **Seleccionar nodo para el siguiente envío:** Tendrán prioridad los nodos vecinos que todavía no se les ha enviado mensajes. Si se les hubiera enviado ya a todos los nodos vecinos mensajes, será seleccionado el primer nodo vecino que tengamos en la lista.

      Tras ser seleccionado pasará otro filtro o criba, este consiste en una probabilidad para poder enviar. En caso de que falle se selecciona el siguiente nodo vecino al que no se le haya enviado información. Si en la lista ninguno supera con éxito la probabilidad, no se prepara el envío hasta el siguiente periodo.

    - **Preparar siguiente escaneado de vecinos:** Cuando no hay aún vecinos en la lista del nodo, o se ha perdido el enlace con algún vecino se prepara un escaneado. Este escaneo de vecinos se realiza tras el anunciar y recibir del siguiente periodo.

- **Descansar:** Cuando se han realizado las tareas de mantenimiento y usuario el nodo puede entrar en modo dormir. Este despierta cuando comienza el siguiente periodo o hay un envío que realizar.

5.13.2 Transiciones de los estados del nodo

Ahora explicaremos cuales son las transiciones de los estados del nodo. El estado que más prioridad tiene es el de anunciar y recibir. Anunciar y recibir que se ejecute al inicio de cada periodo, ya que habrá nodos que traten de enviarle mensajes.

Por otro lado, enviar mensajes a un nodo tiene prioridad sobre cualquier tarea de mantenimiento o usuario, intentando que el nodo esté preparado para enviar. Puesto que el nodo receptor tiene un tiempo estipulado para recibir y no estará esperando a los nodos que quieran enviarle mensajes.

En el diagrama de estado de la Ilustración 20, se aprecia que desde cualquier estado se puede ir al estado anunciar, cuando es tiempo para anunciar. Debido a que el sistema no es *preemptivo*, es decir, no desaloja tareas, aunque estas tengan menos prioridad. Antes de realizar un envío o escanear determinaremos cual es la duración máxima para no desplazar en el tiempo el estado anunciar.

El retraso en el tiempo de anunciar afecta a los nodos vecinos, teniendo los vecinos que escanear de nuevo al perder la sincronización.

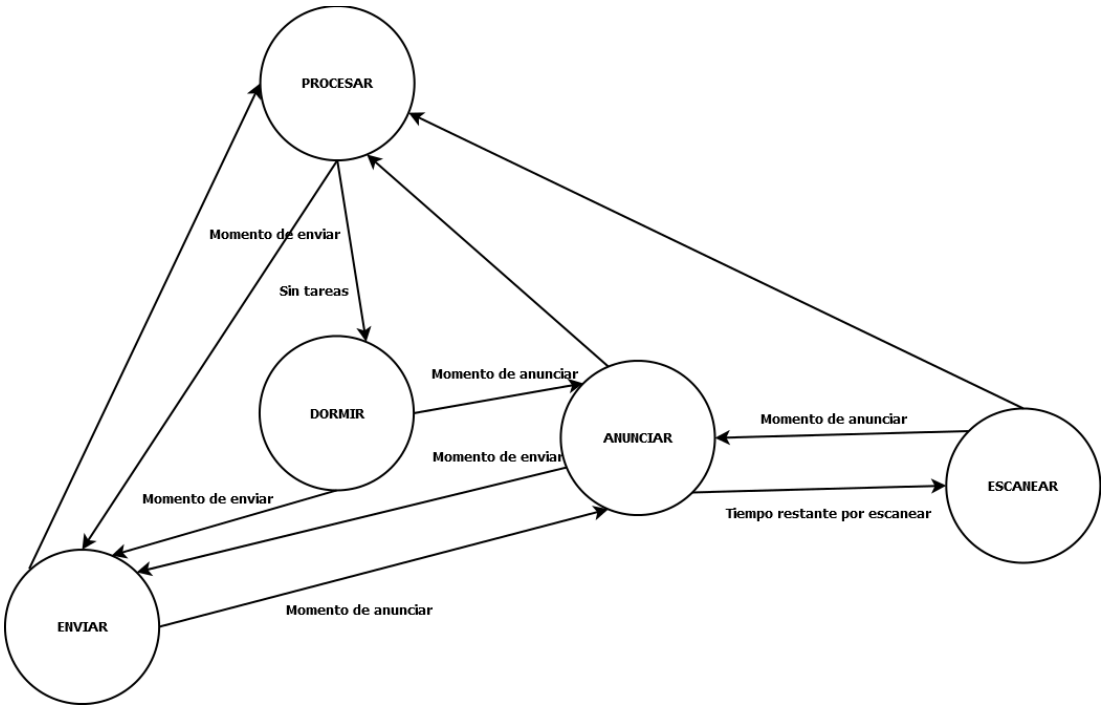

Ilustración 20.-Diagrama de estados del nodo

En la Tabla 8 mostramos de forma resumida cuando ocurren las transiciones entre estados.

Origen\Destino	Anunciar	Escanear	Enviar	Procesar	Dormir
Anunciar		En caso de que se haya establecido en el *estado procesar* la tarea de escanear.	Enviar al nodo durante el tiempo restante.	En caso de no tener que enviar ni escanear.	
Escanear	Escanea x segundos hasta que hay que anunciar. Tras anunciar, termina de escanear los segundos restantes.				
Enviar	Intenta enviar durante el tiempo de envío o hasta que haya que anunciar.			En caso de no tener que enviar (sólo una vez por periodo como máximo).	
Procesar			Ir a enviar, si es tiempo de envío tras procesar alguna tarea.		Tareas realizadas.
Dormir	Tiempo de anunciar.		Tiempo de enviar.		

Tabla 8.-Transiciones de los estados del nodo

5.13.3 Cálculo del tiempo del periodo del nodo

Antes de explicar y justificar la duración del periodo, vamos a mostrar un escenario entre dos nodos que puede ser problemático. En la Ilustración 21, se muestra el problema en concreto en el segundo periodo. El estado anunciar y recibir tiene la misma duración para ambos nodos, además, también coinciden periódicamente debido.

La solución propuesta se puede ver tanto en el primer ciclo como en el tercer ciclo. Para ello el nodo puede tomar una duración de recepción ‘d’ o ‘2*d’ de forma aleatoria. De modo que no evita que ocurra este escenario, pero si evita que ocurra de forma continuada.

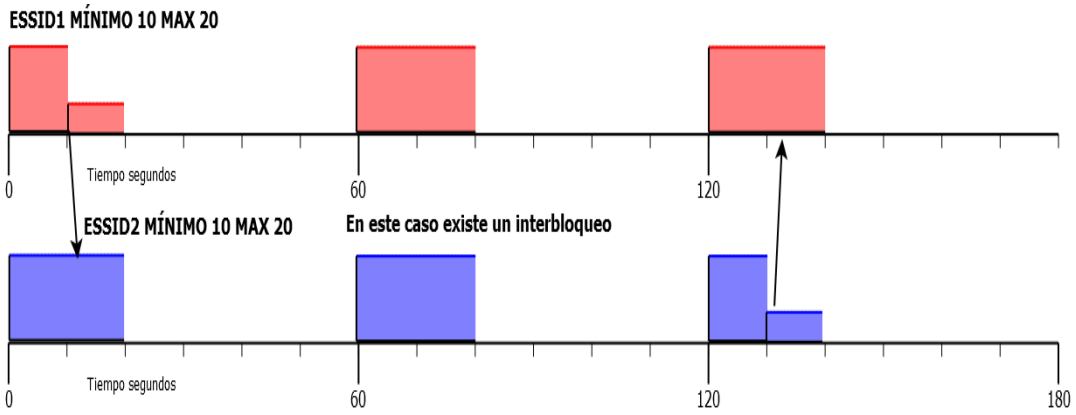

Ilustración 21.-Solución al interbloqueo de dos nodos

La duración del tiempo de anunciar y recibir determina la duración del envío. Hemos decidido que sea igual a la duración mínima de la duración de anunciar y recibir, es decir ‘d’.

Una vez conocidos los tiempos de duración de los estados anunciar/recibir y de enviar; debemos conocer cuánto tiempo requiere ejecutar las tareas de mantenimiento y de usuario en el peor caso.

Para calcular cuánto tiempo es necesario establecer como periodo, hay que sumar el tiempo que se requiere en el peor escenario.

$$tiempo\ mínimo\ del\ periodo\ de\ un\ nodo$$
$$= 2 * duración\ de\ recibir\ y\ anunciar + duración\ de\ enviar$$
$$+ \sum duración\ tareas\ de\ mantenimiento\ en\ el\ peor\ caso$$
$$+ \sum duración\ de\ tareas\ de\ usuario\ en\ el\ peor\ caso$$
$$+ máx(duración\ del\ conjunto\ de\ tareas\ de\ mantenimiento$$
$$\cup\ duración\ de\ conjunto\ de\ tareas\ de\ usuario)$$

Debemos sumar el tiempo de la tarea de mantenimiento o de usuario que dure más tiempo. Ya que el peor caso es que la tarea con mayor duración no pueda ejecutarse debido a que un envío debe realizarse muy próximo al final de la ejecución de la tarea. En la Ilustración 22 se muestra el mismo nodo sin añadir esta duración extra, la tarea en el segundo ciclo queda fuera ya que, si se ejecutara, la posibilidad de enviar se perdería.

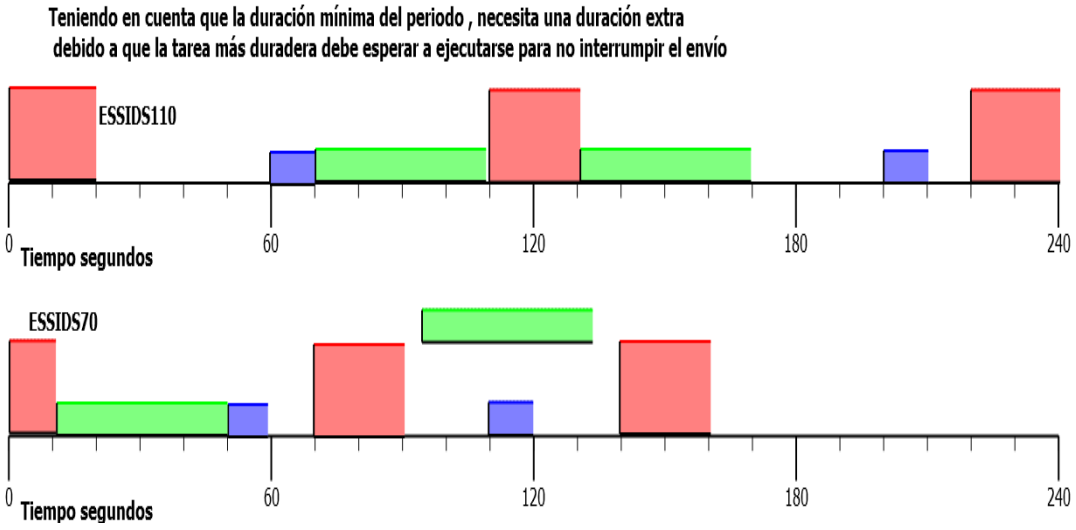

Ilustración 22.-Añadiendo la duración de la tarea más duradera al periodo

5.13.4 Tipos de mensajes

Cada vez que un nodo va a enviar a otro nodo, le envía primero un mensaje de control, y seguidamente le envía mensajes de datos.

- **Mensajes de control:** Controlan la probabilidad de que un nodo vecino pueda conectar y enviar mensajes al nodo receptor. Es necesario para no colapsar la memoria de los nodos. El mensaje de control contiene los siguientes campos:

- **Tasa o *rate*:** Indica la dificultad a superar para conectar con el nodo indicado por el campo id.
- **Id:** Este campo es enviado para que el receptor reconozca a quien pertenece el mensaje de control.
- **Canal o *channel*:** Los mensajes de control pertenecen al canal "_RATE". El mensaje de control se lee en tiempo de tareas de mantenimiento, donde se actualiza la dificultad del nodo indicado por el 'id'.

La dificultad se utiliza cuando el nodo va a seleccionar al nodo vecino al que quiere enviar. Se utiliza una función de aleatoriedad, donde puede suceder desde el número 0 hasta el número indicado en la dificultad. En caso de éxito es decir si sale el 0, esto ocurrirá con una probabilidad de 1/dificultad.

En nuestro caso la dificultad la definimos por el número de mensajes recibidos/número de mensajes eliminados.

Ejemplo de mensaje de control

{"rate":1,"channel":"_RATE","id":"BN581722S60"}

- **Mensajes de datos:** Se caracterizan por tener asociados un canal, datos, la ruta de los nodos por los que ha ido pasando y el número de secuencia.
    - **Canal o *Channel*:** es un metadato para los nodos que estén interesados en los mensajes del canal.
    - **Datos o data:** dentro va la información útil.

- **Ids:** Contiene los nodos por los que ha ido pasando el mensaje, es necesario para que los nodos no tengan que almacenar si el mensaje ha estado previamente en su posesión. Cuando se revisan los mensajes recibidos, si el id propio ya está contenido en el vector de 'ids', implica que el mensaje ya ha estado en el nodo, con lo cual se descarta el mensaje. En caso de que no se encuentre el id, el mensaje pasa a la cola de mensajes sin leer y se le añade el 'id' del nodo al vector de 'ids' del mensaje.

  El nodo origen es el primer nodo en el vector de identificadores.

- **Secuencia:** Es un número que indica el orden de creación del mensaje, en el nodo origen para el canal.

- **Numero de identificadores:** Necesario para facilitar la decodificación del vector de identificadores.

Ejemplos de mensajes de datos donde se informa de algunas estadísticas del nodo.

*{"**data**":{"n_try_connections":3100,"n_failed_connections":107,"n_scanned":2,"n_messages_sent":8883,"n_messages_received":8982,"n_messages_repeatly":0,"n_messages_wasHere":2794,"n_messages_removed":5890,"n_sleeps":6201},"**ids**":["BN1252024S60"],"**channel**":"_STATS","**sequence**":2797,"**n_ids**":1}*

*{"**data**":{"n_try_connections":3982,"n_failed_connections":69,"n_scanned":19,"n_messages_sent":11411,"n_messages_received":11301,"n_messages_repeatly":0,"n_messages_wasHere":3906,"n_messages_removed":7498,"n_sleeps":7853},"**ids**":["BN581722S60","BN1252024S60"],"**channel**":"_STATS","sequence":3912,"**n_ids**":2}*

5.13.5 Tiempo teórico de entrega de mensajes

En una red de sensores es complejo medir la latencia por diferentes factores movilidad de los nodos, tipo de enrutamiento, interferencias en el medio y rotura del enlace por el fallo o caída de algún nodo crítico en la ruta.

En el sistema desarrollado los factores que más influyen en la latencia desde que un mensaje se origina hasta que llega a un nodo interesado o nodo destino, son los siguientes:

- **Probabilidad de acceder al nodo adecuado en la ruta:** En el peor caso la probabilidad de fallo seguirá una distribución de Bernoulli o también conocida como distribución dicotómica. Así que suponiendo que existen un conjunto de nodos de los cuales todos ellos excepto uno no están en la ruta, la probabilidad de éxito es de 1 / nº de nodos vecinos

- **Probabilidad de acceso al nodo:** Bajo condiciones normales la probabilidad de acceso al nodo va a depender del número de nodos que tratan de acceder a ese nodo en concreto. Suponiendo que el nodo en concreto tiene m nodos que tratan de acceder a él, la probabilidad de éxito para acceder a ese nodo en concreto es de 1/m nodos que tratan de acceder al nodo concreto.

  Con estas dos probabilidades de éxito se deben combinar dando lugar a la probabilidad de entregar el mensaje al único nodo válido en la ruta. Esta probabilidad sería 1/ (nº de nodos vecinos del nodo emisor*m nodos que tratan de acceder al nodo crucial en la ruta)

  Esto puede ocurrir en la siguiente disposición de una región de sensores y actuadores para un salto de la ruta.

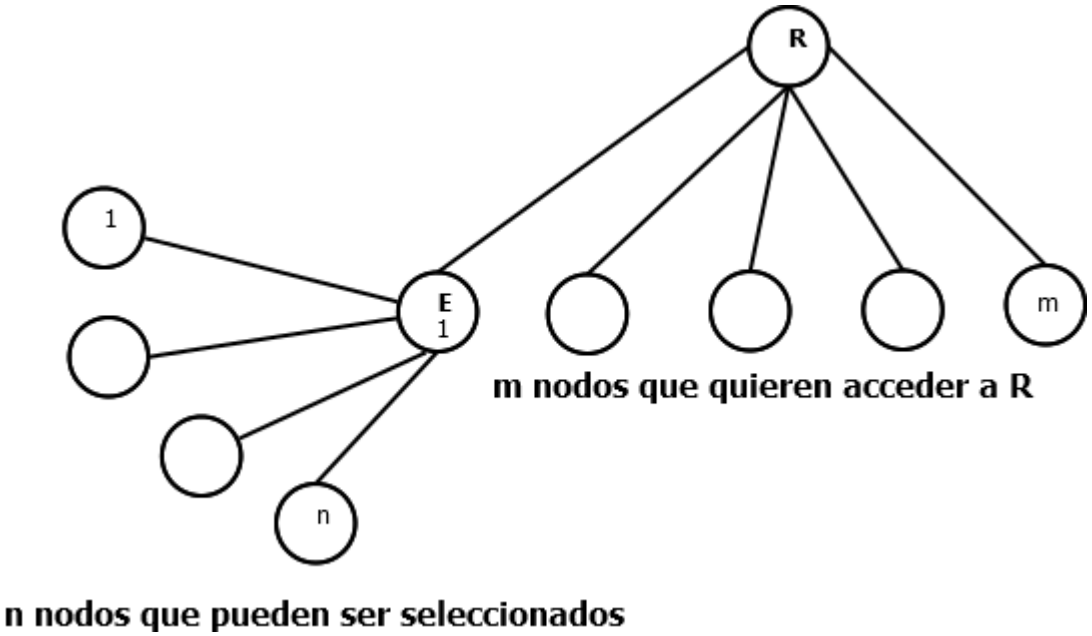

Ilustración 23.-Peor caso para entregar un mensaje al nodo adecuado

Suponiendo que todos los nodos tengan el máximo tiempo de periodo y que cada salto en la ruta tiene presenta un escenario similar al de la Ilustración 23.

$$\sum_{i=0}^{n\ saltos} duración\ máxima\ del\ periodo * n\ vecinos\ en\ i * m\ vecinos\ en\ i + 1$$

En esta fórmula suponemos que los mensajes entrantes están disponibles para reenviarlos al siguiente periodo.

Como vemos la latencia es excesiva al haber utilizado gossiping controlado, como protocolo de enrutamiento. Los otros protocolos que hemos visto tendrían ese problema al existir diferentes actuadores (nodos interesados), ya que implicaría tener que enviar a todos los puntos interesados. En cuanto a SPIN al ser en tres pasos aumentaría su latencia al multiplicarse la necesidad de comunicarse tres veces con cada nodo.

Para tratar de reducir la latencia no debemos densificar la región de nodos y transmitir pocos mensajes para no colapsar la red.

5.13.6 Gasto energético teórico

Usualmente se ofrece la intensidad eléctrica media por segundo. Además, en sistemas donde hay una fracción del tiempo activo y otra fracción del tiempo desactivado, se calcula la media de la intensidad por segundo de la siguiente forma (26).

$$Intensidad\ media\ por\ segundo = \frac{Intensidad\ media\ activo * duración\ activo}{periodo} + \left(1 - duración\frac{activo}{periodo}\right) * Intensidad\ media\ desactivado$$

En nuestro desarrollo el consumo medio por segundo estando activo va a depender de:

- **Intensidad media por segundo recibiendo mensajes:** En el caso de recibir mensajes el ESP8226 utiliza una corriente media de 50 miliamperios por segundo. La duración de recibir puede ser el doble por el mecanismo desarrollado para evitar el bloqueo muto. Entonces la duración media de recibir es de tres medios la duración de recibir mínima.

$$Intensidad\ media\ por\ segundos\ recibiendo\ mensajes = Intensidad\ media\ por\ segundo\ recibiendo * duración\ recibir * 1.5/Periodo = 56 * 1{,}5 * duración\ de\ recibir\ mínima/Periodo = 84mA * duración\ de\ recibir\ mínima/Periodo$$

- **Intensidad media por segundo transmitiendo mensajes:** En el caso del ESP8226 y utilizando por defecto 802.11.n, consume 170 por segundo cuando está transmitiendo. La duración del envío es la misma duración que la duración de recibir mínima.

$$Intensidad\ media\ por\ segundos\ transmitiendo\ mensajes = Intensidad\ media\ por\ segundo\ transmitiendo * duración\ en\ segundos\ del\ envío/Periodo = 170mA * duración\ de\ recibir\ mínima/Periodo$$

- **Intensidad media por segundo escaneando:** Cuando escaneamos, consumimos 170 miliamperios por segundo. La duración del escaneo es igual al tiempo de duración máximo para el periodo de cualquier nodo de la red, para asegurar que encontramos a todos los vecinos en el entorno. Sin embargo, el número de veces que se realiza por periodo puede variar. Existe un escaneo inicial, y después en función de si se rompe algún enlace con algún nodo vecino. Debemos asociar a esta intensidad media por segundo escaneando, una probabilidad de rotura de enlace.

$$Intensidad\ media\ por\ segundo\ escaneando = Intensidad\ media\ por\ segundo\ al\ escanear * (duración\ del\ periodo\ máximo)/(Periodo * Nº\ de\ Periodos\ medio\ hasta\ rotura\ de\ enlace) = 170mA * (duración\ del\ periodo\ máximo)/(Periodo * Nº\ de\ Periodos\ medio\ hasta\ rotura\ de\ enlace)$$

- **Intensidad media por segundo realizando tareas de mantenimiento y usuario:** Este consumo depende de las tareas de usuario y su intensidad media por segundo puede oscilar entre 20 miliamperios si no se utiliza WiFi, hasta 170 miliamperios si se utiliza. Tomando el término medio 85 miliamperios.

$$Intensidad\ media\ por\ segundo\ realizando\ tareas\ de\ mantenimiento\ y\ usuario = intensidad\ media\ por\ segundo\ necesaria\ para\ tareas * duración\ de\ las\ \frac{tareas}{Periodo} = 85mA * duración\ de\ las\ tareas/Periodo$$

Por otra parte, debemos calcular que fracción por segundo medio va estar dormido el nodo.

1. **Vamos a exponer la fórmula para el cálculo de la fracción por segundo medio del estado dormido**.

$$duración\ media\ por\ segundo\ de\ dormir = \left(1 - \left(1{,}5 * duración\ de\ recibir\ \frac{mínima}{Periodo} + duración\ de\ recibir\ \frac{mínima}{Periodo} + \frac{Periodo}{Periodo\ máximo * nº\ de\ Periodos\ medio\ hasta\ la\ rotura\ del\ enlace} + duración\ del\ tiempo\ de\ tareas\ de\ mantenimiento\ y\ usuario/Periodo\right)\right)$$

2. **Intensidad media por segundo durmiendo:** El nodo en modo dormido y que puede despertarse tras un tiempo establecido, necesita 0,015 miliamperios.

$$Intensidad\ media\ por\ segundo\ durmiendo = Intensidad\ media\ \ por\ segundo\ dormido * duración\ media\ por\ segundo\ de\ dormir = 0{,}015\text{mA} * duración\ media\ por\ segundo\ de\ dormir$$

Teniendo las fórmulas necesarias de ambas partes, la intensidad media por segundo para el sistema activo y por otra parte para el sistema inactivo, podemos expresar la fórmula expandida, pero vamos a utilizar abreviaturas para que no se haga muy extensa.

$$Intensidad\ media\ por\ segundo = 84mA * \frac{d}{p} + 170mA * \frac{d}{p} + 170mA * \frac{P}{p * r} + 85mA * \frac{T}{p} + \left[1 - \left[\left(1{,}5 * \frac{d}{p}\right) + \left(\frac{d}{p}\right) + \left(\frac{P}{p * r}\right) + \left(\frac{T}{p}\right)\right]\right] * 0{,}015mA$$

Donde los parámetros son:

$$d = duración\ mínima\ de\ recibir\ en\ segundos\ por\ periodo$$

$$p = duración\ del\ periodo\ del\ nodo\ en\ segundos$$

$$P = duración\ del\ periodo\ máximo\ para\ cualquier\ nodo\ en\ la\ red\ en\ segundos$$

$$r = nº\ de\ roturas\ de\ enlaces\ medio\ por\ periodo$$

$$T = duración\ de\ las\ tareas\ de\ mantenimiento\ y\ de\ usuarios\ en\ segundos\ por\ periodo$$

En base a estos cinco parámetros, podemos calcular la intensidad media por segundo del nodo.

5.13.7 Diagrama de clases

En la Ilustración 24 mostramos el diagrama de las clases principales:

- **Clase 'APs'**: Se encarga de gestionar la lista de vecinos del nodo.
- **Clase 'broadcastNode'**: Gestiona el radio WiFi del nodo, además de gestionar las conexiones. Para gestionar las conexiones tiene un servidor TCP con un puerto por igual para todos los nodos en la red. Por otro lado, dispone de una lista de clientes TCP. También contiene la información del ESSID del nodo.
- **Clase 'messageBroker'**: Gestiona la cola de mensajes sin revisar, la cola de mensajes listos para leer, la cola de mensajes para enviar.
- **Clase 'SchedulerNode'**: Es la más compleja porque se encarga de las transiciones de estado y de ejecutar las funciones del estado, para ello se provee de una lista de nodos vecinos, dispone de las colas de mensajes y de las capacidades del radio WiFi. Además, tiene una lista de tareas, tareas de mantenimiento y usuario. Estas tareas se ejecutan de forma no *preemptiva* dentro del periodo.

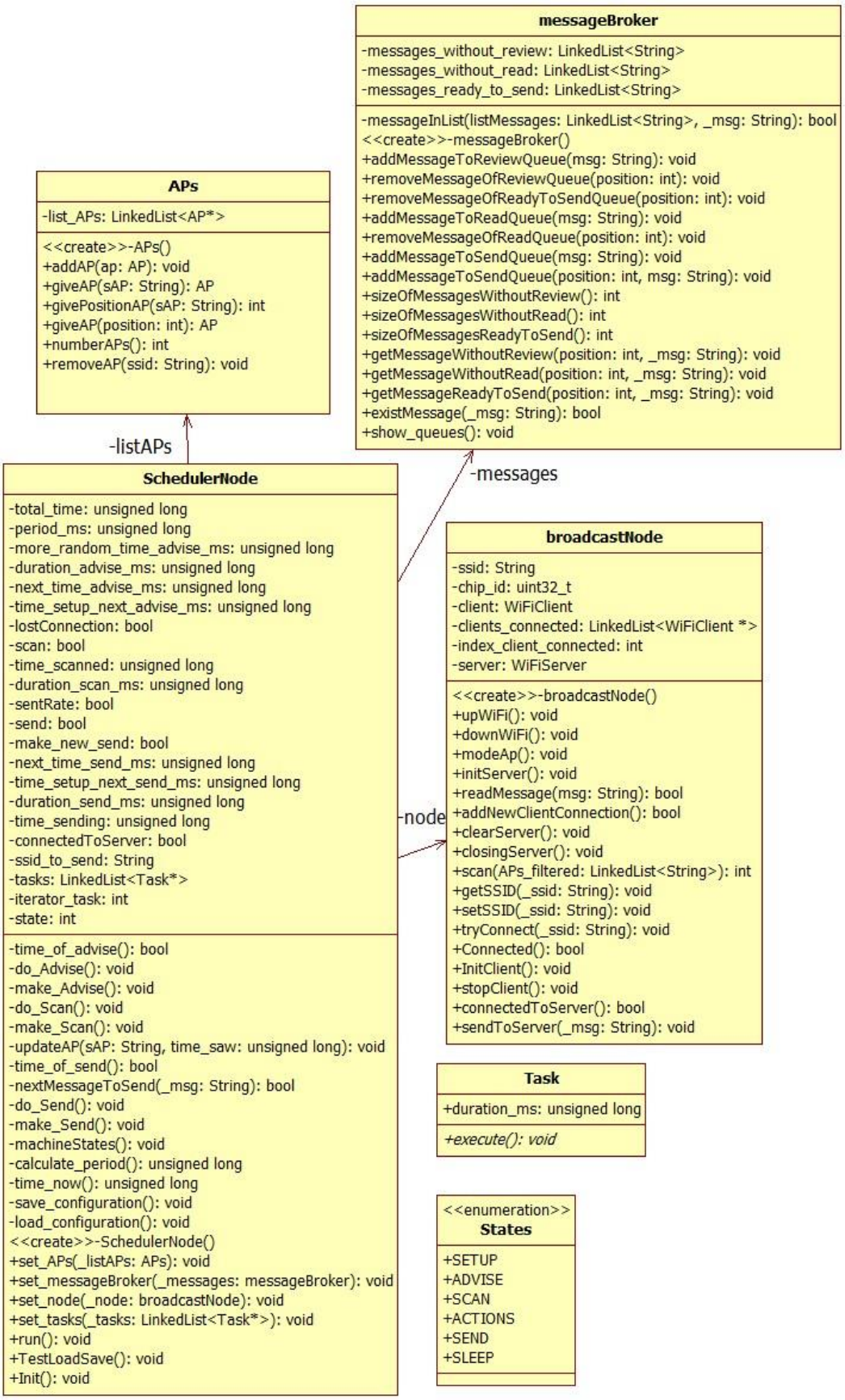

Ilustración 24.-Diagrama de las clases principal

5.13.8 Diagrama de secuencias

Presentamos los diagramas de secuencias más interesantes para el desarrollo, los cuales son:

- **Realizar anuncio y recibir mensajes**

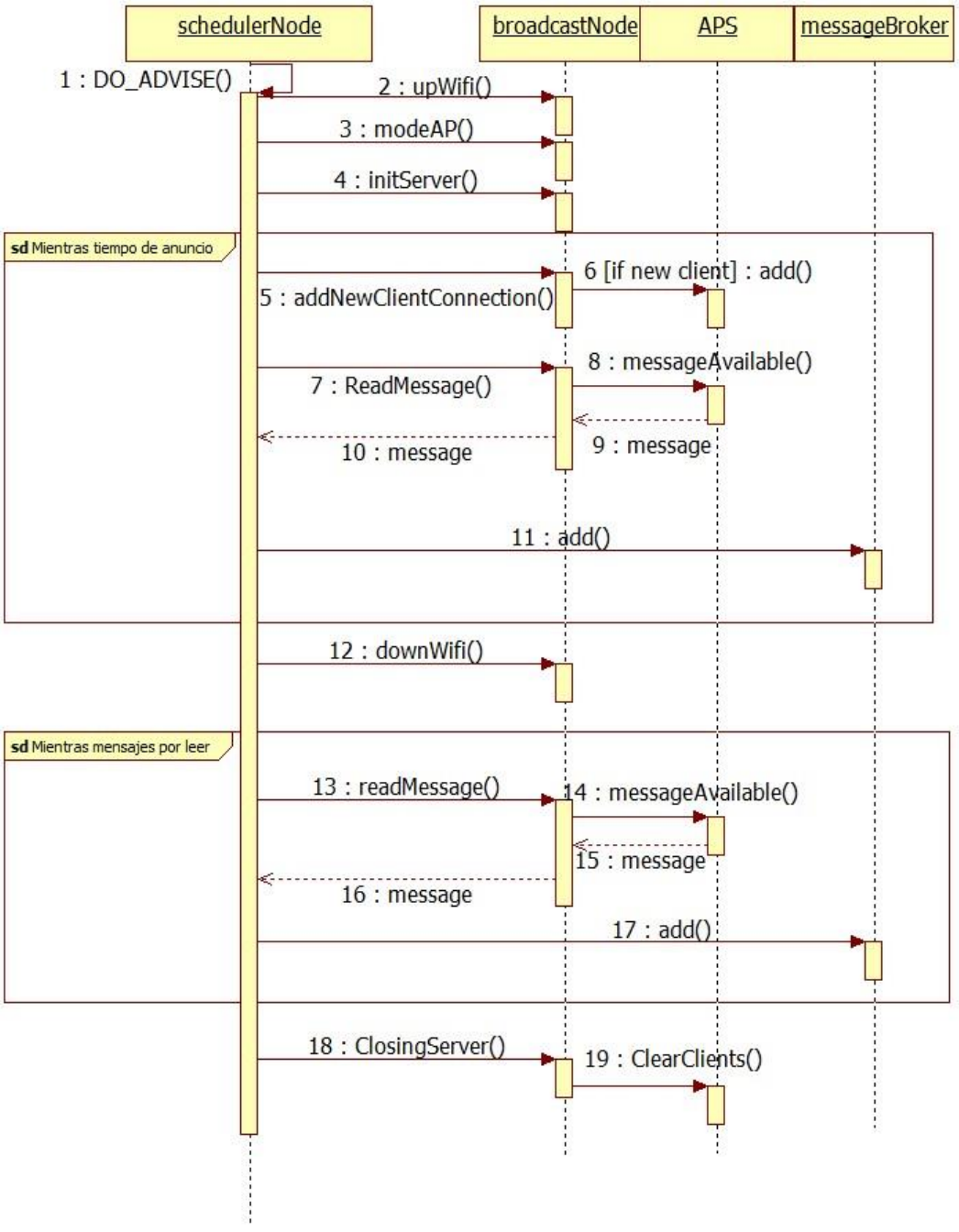

Ilustración 25.-Diagrama de secuencias (Anunciar y recibir mensajes)

El tiempo de anuncio puede ser la duración de recibir, o dos veces la duración de recibir, con un cincuenta por ciento de probabilidad para ambas.

- **Encontrar vecinos y recuperar enlaces rotos**

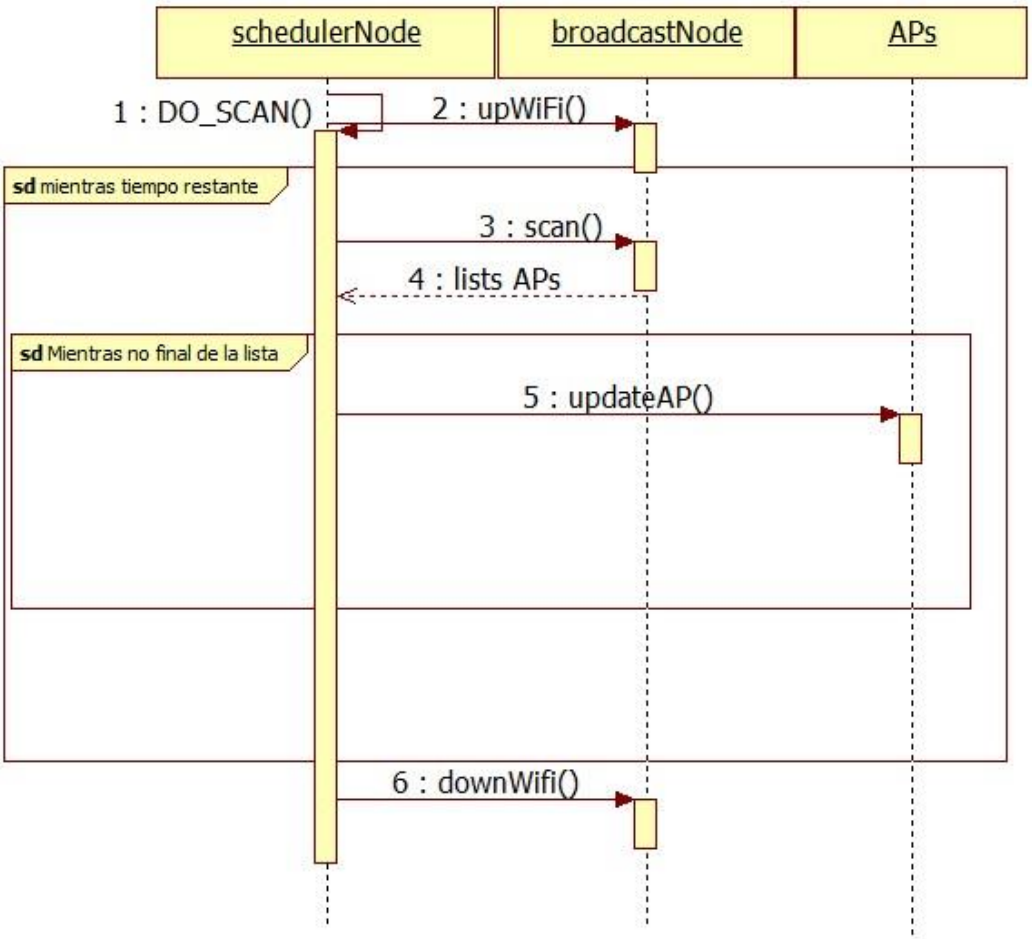

Ilustración 26.-Diagrama de secuencias Buscar nodos

El tiempo restante es necesario para interrumpir la tarea de escanear, cuando vaya a ocurrir el estado anunciar y recibir. Si quedara más tiempo por escanear, tras terminar el recibir y anunciar, volveríamos a escanear con el tiempo restante.

Es interesante destacar que para registrar el tiempo en el que fue visto el vecino, está calculado mediante la siguiente media:

$$tiempo\ intermedio = (tiempo\ antes\ de\ escanear + tiempo\ despues\ de\ escanear)/2$$

- **Enviar mensajes**

Es solamente interrumpido por anunciar y recibir mensajes, al igual que en el estado escanear. Para ello, calculamos previamente cuanto tiempo puede estar en el estado. De modo que, no interfiere en la desincronización. La desincronización puede ocurrir cuando las funciones del estado anunciar y recibir se ejecuten fuera del plazo.

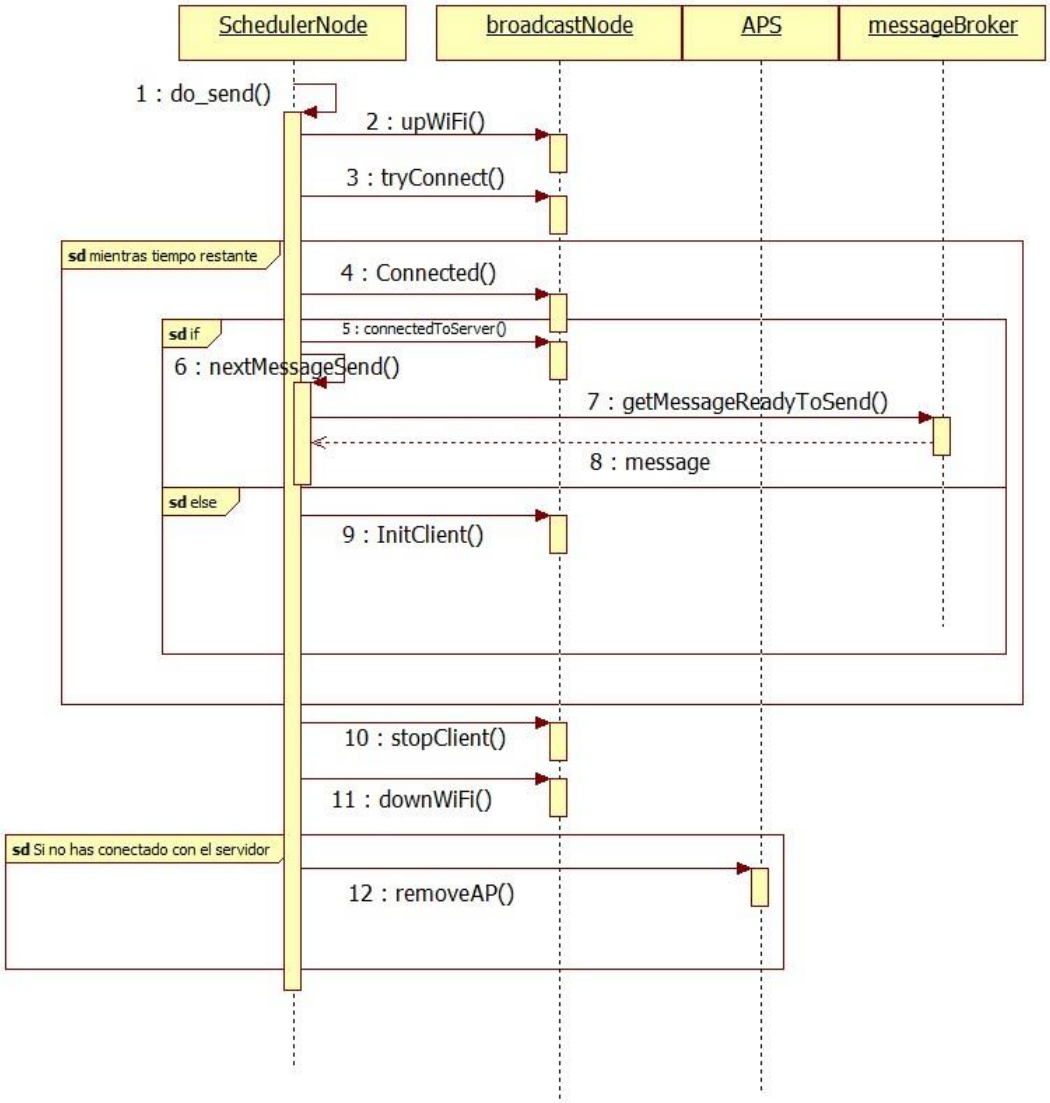

Ilustración 27.-Diagrama de secuencias Enviar mensajes

5.13.9 Tareas de mantenimiento

Son las tareas necesarias para eliminar mensajes que no son necesarios mantener, revisar los mensajes entrantes, actualizar y crear el mensaje de control.

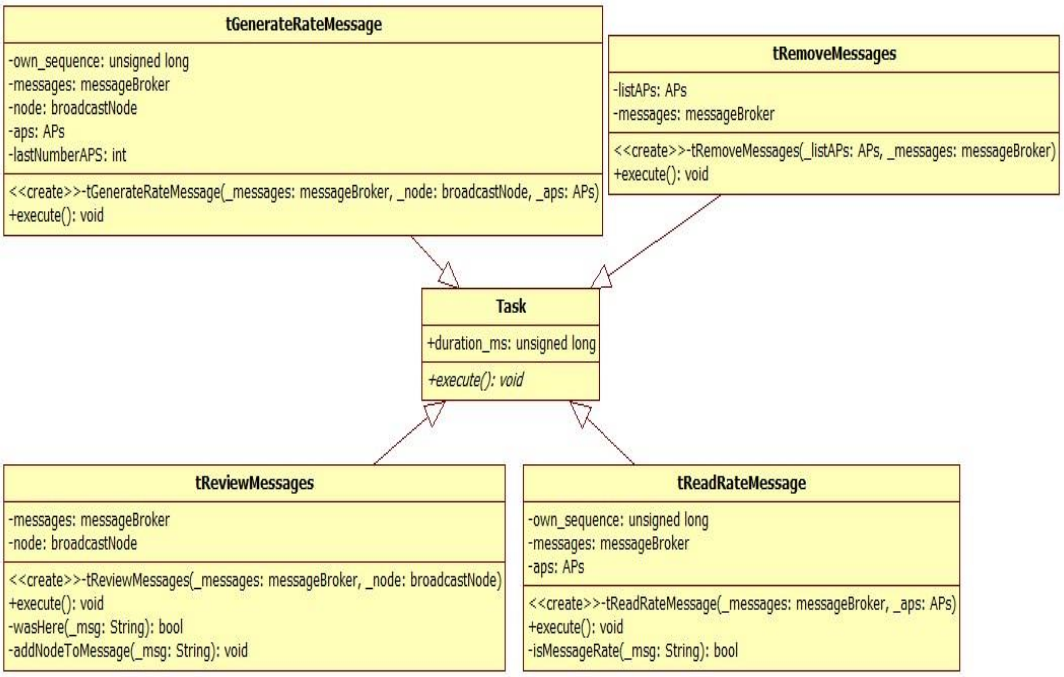

Ilustración 28.-Diagrama de clases de mantenimiento del nodo

Las tareas de usuario y mantenimiento heredan de la clase 'Task', donde la duración en milisegundos es importante para calcular el periodo mínimo. Además, también es necesario para saber si se puede ejecutar, ya que puede interferir en el plazo de un envío.

5.14 Implementación

Podéis descargar, ver y contribuir al proyecto en GitHub. Se encuentra en ***https://github.com/enriquecml/SnorlaxMesh*** .

Se requieren las siguientes librerías de terceros:

Para gestionar cadenas en formato JSON:

https://github.com/bblanchon/ArduinoJson

Plantilla de lista enlazada:

https://github.com/ivanseidel/LinkedList

Generador de números aleatorios:

https://github.com/marvinroger/ESP8266TrueRandom

5.15 Manual de usuario desarrollador

Entregamos un componente para desarrollar redes multisalto, que generen pocos eventos en los nodos.

La interfaz del componente la hemos llamado ‘SnorlaxMesh’, en castellano contracción de *malla roncar relajar*.

Este componente actualmente realiza lo siguiente:

- Crea los objetos necesarios para el planificador del nodo.
    - Cola de mensajes
    - Objeto encargado del radio WiFi y las gestiones de las conexiones
    - Lista de vecinos
    - Una lista de tareas, añadiendo por defecto las tareas de mantenimiento.
- Ejecuta el planificador del nodo

5.15.1 Crear tarea de usuario

La tarea de usuario debe heredar de la clase ‘Task’.

Debe medir por seguridad los milisegundos que puede tardar la tarea en el peor caso, para garantizar el normo funcionamiento del sistema. Es tiempo srá asignado al atributo ‘duration_ms’.

Debe implementar el método ‘execute()’ sin argumentos. Puedes crear métodos, específicos de la tarea y pasar por referencia al constructor, la cola de mensajes, el radio WiFi, la lista de los nodos vecinos, u otros objetos desarrollados por el usuario o terceros desarrolladores.

5.15.1.1 Advertencias

A la hora de utilizar la cola de mensajes de envío, por alguna razón de la aplicación a desarrollar. Debemos tener especial cuidado con desplazar el mensaje que está en la

posición 0, ya que en nuestro sistema siempre se envía en cada conexión el mensaje de control, el cual debe permanecer en la posición 0.

Cuando desee utilizar el radio WiFi, por ejemplo, para enviar información a Internet a través de un router.

Hay que utilizar el objeto 'broadcastNode' que se encuentra en el componente 'SnorlaxMesh'. En ese objeto utilizamos los métodos 'upWiFi()' y 'downWiFi()' al finalizar.

En el caso de que necesiten alguna librería que utilice los métodos WiFi, bien del sdk base o de Arduino, es imprescindible que antes de ejecutar métodos de la librería, se levante el WiFi mediante 'upWiFi()'; y que al finalizar la ejecución de los métodos de la librería, se añada detrás 'downWiFi()'. Estos métodos son necesarios para que el esp8226 consuma 20 miliamperios.

5.15.2 Añadirla tarea al planificador

Aún no hemos desarrollado la API completa para acceder a los objetos creados por el componente 'SnorlaxMesh'. Por el momento, la tarea de usuario debe añadirse a la lista de tareas del componente, mediante el operador *new de c++*.

Como ejemplo de tarea desarrollada podéis ver en GitHub la tarea para generar mensajes de estadísticas y leer mensajes de estadísticas.

https://github.com/enriquecml/SnorlaxMesh/blob/master/tGenerateStatsMessage.h

https://github.com/enriquecml/SnorlaxMesh/blob/master/tGenerateStatsMessage.cpp

https://github.com/enriquecml/SnorlaxMesh/blob/master/tReadStatsMessage.h

https://github.com/enriquecml/SnorlaxMesh/blob/master/tReadStatsMessage.cpp

Además del trozo de código añadido en el componente 'SnorlaxMesh'.

https://github.com/enriquecml/SnorlaxMesh/blob/master/SnorlaxMesh.h

https://github.com/enriquecml/SnorlaxMesh/blob/master/SnorlaxMesh.cpp

6 RESULTADOS Y DISCUSIÓN

6.1 Experimento final

6.1.1 Motivación

En este último experimento pretendemos observar el comportamiento de una red de nodos ESP8226, durante un periodo de tiempo en un escenario real. Con el objetivo de evaluar el progreso del desarrollo realizado y detectar posibles deficiencias.

6.1.2 Escenario

Hemos escogido como lugar unas fincas de viñedos situadas en la periferia de Almendralejo. El experimento se realizó el día 2 de septiembre de 2017. Comenzando a las 19:45 y finalizando a las 21:00. Había pequeñas rachas de viento y una temperatura de unos 30º.

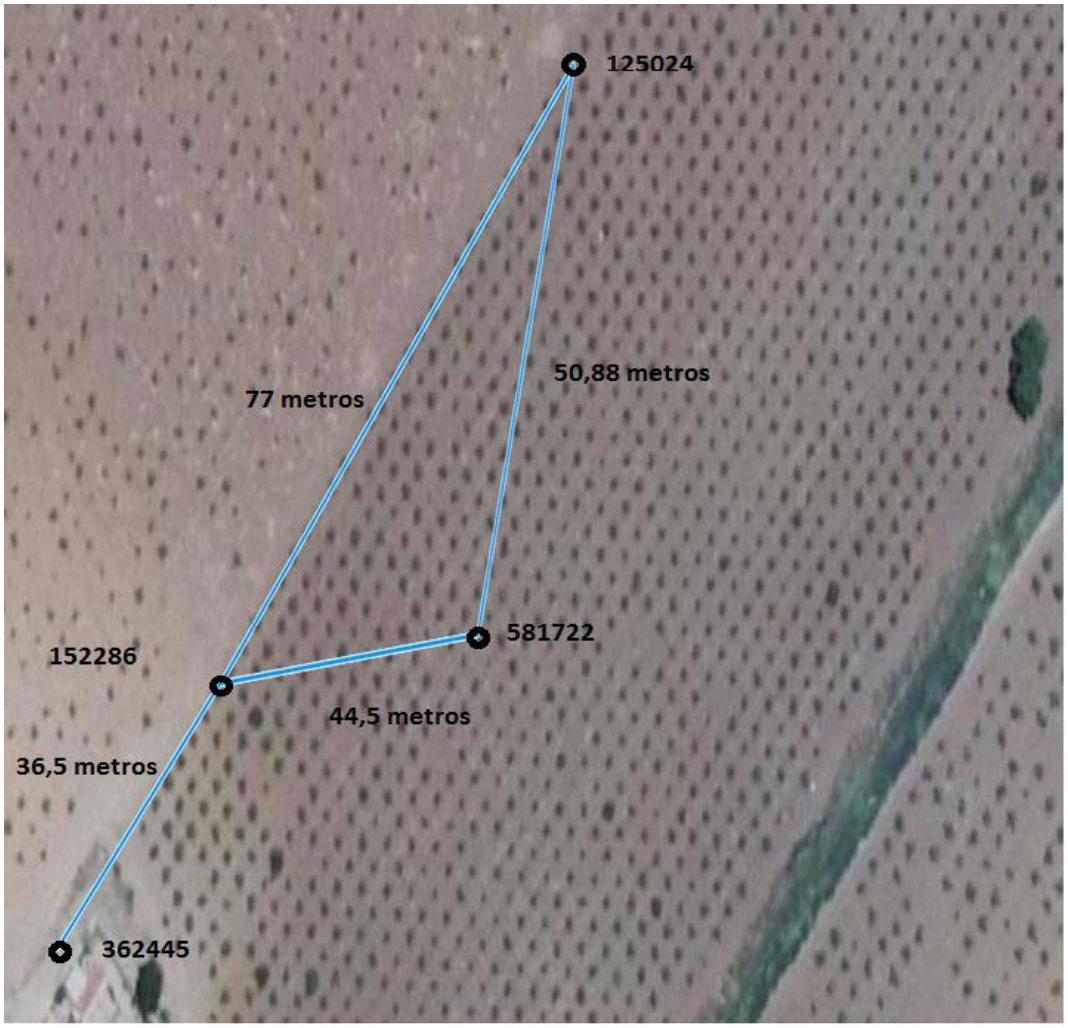

Ilustración 29.-Despliegue de los nodos en una finca para el experimento

6.1.3 Materiales

- Cuatro placas de desarrollo ESP8226.
- Tres baterías portátiles de 5000 mAh cargadas
- Cuatro conectores USB-micro-usb-c
- Un ordenador portátil(Sumidero)

6.1.4 Parámetros de los nodos

Hemos utilizado los mismos parámetros para todos los nodos.

d = duración mínima de recibir en segundos por periodo = 6s

p = duración del periodo del nodo en segundos = 60s

P = duración del periodo máximo para cualquier nodo en la red en segundos = 60s

r=promedio de rotura del enlace por periodo = Desconocido

T = duración de las tareas en segundos por periodo

$T < p - 3 * d$; $T<60s-18s$; $T < 48$ s

Algunos parámetros son desconocidos. Porque aún no se ha calculado los tiempos de las tareas, ni se han hecho pruebas para ver si existe una correlación entre la densidad de nodos en una región, la movilidad de los nodos y la rotura del enlace.

Pero sabemos que con sesenta segundos de periodo da tiempo a ejecutarse las pocas tareas.

6.1.5 Sucesos

19:45 Colocando nodos a nivel del suelo

20:15 Colocados encima de la cepa a medio metro

20:30 Recolocación del nodo enlace porque se ha caído de la cepa debido al viento

20:45 Subimos el nodo receptor a un metro 1,70 de altura

21:00 Nodos recogidos

6.2 Resultados

Los mensajes de estadísticas extraídos son procedentes del nodo 362445, nodo que estaba conectado al portátil.

Hemos decidido agrupar los mensajes recibidos, en base al nodo origen del mensaje de estadística y ordenados por la secuencia.

La mayoría de los mensajes han sido recibido tras subir los nodos a la altura de la cepa y tras volver a colocar el nodo 152286, el cual se había caído tras una racha de viento. También hemos decidido omitir el campo número de escaneos realizados, debido a que, en el experimento, han existido bastantes minutos sin comunicación. Esto se debe a que no se ha empezado por una altura adecuada para la colocación de los nodos.

Enviados	Recibidos	Repetidos	Han estado aquí	Eliminados	Veces dormido	Secuencia	Pasa por 1
0	0	0	0	0	0	0	152286
5	5	0	1	1	10	1	152286
6	9	0	1	2	14	2	152286

Tabla 9.-Mensajes recibidos con origen en nodo 125024

Enviados	Recibidos	Repetidos	Han estado aquí	Eliminados	Veces dormido	Secuencia
0	0	0	0	0	0	0
2	0	0	0	1	13	1
21	19	0	1	11	39	3

Tabla 10.-Mensajes recibidos con origen en nodo 152286

Enviados	Recibidos	Repetidos	Han estado aquí	Eliminados	Veces dormido	Secuencia	Id 1	Id 2
0	0	0	0	0	0	0	1252024	152286
2	3	0	1	1	7	1	1252024	152286
11	9	0	3	7	15	4	1252024	152286
13	9	0	3	8	17	5	1252024	152286
15	9	0	3	9	19	6	1252024	152286
17	9	0	3	10	21	7	1252024	152286
19	9	0	3	11	23	8	1252024	152286
21	18	0	9	12	25	9	1252024	152286

27	18	0	9	16	29	11	1252024	152286
30	18	0	9	17	38	12	1252024	152286
33	21	0	10	19	47	13	1252024	152286
35	21	0	10	20	50	14	1252024	152286

Tabla 11.-Mensajes recibidos con origen en nodo 581722

Enviados	Recibidos	Repetidos	Han estado aquí	Eliminados	Veces dormido	Secuencia
0	0	0	0	0	0	0
18	9	0	0	8	49	1
32	24	0	1	9	57	2

Tabla 12.-Mensajes generados por el nodo sumidero

6.3 Discusión

Todos los mensajes han sido recibidos a través del enlace sumidero y el nodo 152286, con una distancia entre ambos de 36,5 metros.

La mayoría de los mensajes recibidos han sido del nodo 581722. Estos mensajes han atravesado los nodos (1252024,152286).

Debido a los cambios sufridos por los sucesos ocurridos, se han liberado algunos mensajes antes de llegar a su destino. Esto es así, para liberar memoria cuando ya se les ha enviado a todos los vecinos que tenga en la lista en el momento de eliminar el mensaje. Como el nodo 152286 se cayó, es posible que se hayan perdido secuencias tanto del propio nodo, como algunos mensajes que estuvieran en el nodo.

Se aprecia que en la Tabla 10 y en la Tabla 11 se han perdido algunas secuencias.

A pesar de haber utilizado un protocolo de enrutamiento que genera bastantes mensajes duplicados por la red, se han perdido algunos mensajes. Lo que nos lleva a pensar que la responsabilidad para controlar la recuperación de mensajes perdidos recae en el nivel de aplicación del nodo o tareas del nodo.

Por otra parte, utilizar *flooding* de forma controlada (*gossiping*) no garantiza que la red sea más robusta Pues si existe algún nodo esencial en una ruta crítica y este sale de la red, se perderán los mensajes hasta que se recuperara el camino. Pero no solo pasaría con *flooding*, sino que el problema es inherente a la topología de la red. Es necesario que el nivel de aplicación se haga responsable de reenviar los mensajes, aunque, por otra parte, otro protocolo de enrutamiento permitiría un mejor

rendimiento con regiones más densas de nodo. Si hubiera regiones más densas de nodos, la probabilidad de que hubiera nodos esenciales en una ruta crítica sería menor.

7 CONCLUSIONES

El extracto positivo del proyecto realizado es la viabilidad de utilizar *SoCs* de bajo presupuesto para realizar prototipos de WSAN. Por otra parte, este trabajo reafirma algunas conclusiones expuesta en otros proyectos y artículos:

- El peso de la tecnología empleada para las comunicaciones inalámbricas repercute en la vida útil de la red.
- La existencia de un compromiso, entre la facilidad de despliegue de los nodos que proporciona una estructura descentralizada de la red, y la eficiencia energética que proporciona una estructura jerárquica de la red.
- Los protocolos de enrutamiento en WSN no son eficaces para WSAN, ya que en WSAN hay múltiples destinos.

Además, la influencia que tiene el hardware del nodo sobre los otros planos del sistema WSAN, como el sistema de comunicación y el ámbito de aplicación.

En aplicaciones que requieran inmediatez, se necesita que la información de los eventos transcurra en pocos saltos para disminuir la latencia. Esto se consigue con módulos inalámbricos de largo alcance.

Actualmente están empezando a aparecer en el mercado placas de desarrollo con módulos inalámbricos de largo alcance o incluso que conectan directamente a Internet por precios muy competitivos.

La entrada de hardware con más capacidades computacionales y de comunicación abrirá una puerta a nuevos retos y nuevas formas de administrar los entornos inteligentes.

8 IDEAS Y FUTUROS TRABAJOS

Tras las conclusiones pensamos que estas ideas pueden ser interesantes.

Ideas sobre la mejora del componente SnorlaxMesh

- Desarrollar un protocolo de enrutamiento similar a B.A.T.M.A.N , donde se almacene el siguiente salto para entregar un evento, hacia nodos que estén en la ruta de otros nodos interesados.
- Para mejorar la fiabilidad del sistema, realizar un sistema para medir el tiempo de ejecución de las tareas de mantenimiento y usuario.
- Implementar el guardar el estado del nodo y la carga del estado para que se pueda dormir el nodo físicamente. De modo que alargamos la vida útil del nodo.

Debido a la inserción de infraestructuras de telecomunicación para IoT, puede ser de interés investigar sobre:

- Redes de motas que puedan ser reconfiguradas a distancia para lograr ciertos propósitos.
- Sistemas expertos, que puedan trabajar sobre ciertos parámetros de la red de motas para lograr mayor eficiencia.
- Gestión de los nodos a distancia para modelar su funcionamiento, mediante actualización del software residente en el nodo, o desarrollando un intérprete de operaciones en el sistema.

9 REFERENCIAS

1. *Wireless sensor and actor networks: research challenges.* **Akyildiz, Ian F. y Kasimoglu, Ismail H.** s.l. : Elsevier, 2004, Ad hoc networks, Vol. 2, págs. 351-367.

2. *Ubicomp systems at 20: Progress, opportunities, and challenges.* **Caceres, Ramon and Friday, Adrian.** 1, s.l. : IEEE Pervasive Computing, 2012, Vol. 11, págs. 14-21.

3. *The Computer for the 21st Century.* **Weiser, Mark.** 1991.

4. **Wang, Qinghua and Balasingham, Ilangko.** *Wireless sensor networks-an introduction.* s.l. : INTECH Open Access Publisher, 2010.

5. **Fernández ortiz, Julian.** [En línea] 29 de Noviembre de 2016. [Citado el: 28 de Agosto de 2017.] https://pt.slideshare.net/julianfernandezortiz71/red-zigbee-para-la-monitorizacin-de-variables-atmosfricas-y-meteorolgicas.

6. *Types of Memory in Embedded Systems.* **Barr, Michael.** s.l. : Embedded Systems Programming, 1 de Mayo de 2001.

7. *Wireless sensor networks.* **Lewis, Franck L. y others.** s.l. : New York: Wiley, 2004, Smart environments: technologies, protocols, and applications, págs. 11-46.

8. **MARTÍNEZ, Roberto Fernández.** *Redes inalámbricas de sensores: teoría y aplicación práctica.* Madrid, España : Universidad de La Rioja, 2009.

9. **Klues, Kevin.** *Power management in wireless networks.* St.Louis : s.n., 2006.

10. **Elahi, Ata y Gschwender, Adam.** *ZigBee Wireless Sensor and Control Network.* s.l. : Prentice Hall, 2009.

11. **Plataformas Zigbee .** http://plataformaszigbee.blogspot.com.es/. [En línea] 20 de Noviembre de 2012. [Citado el: 29 de Agosto de 2017.] http://plataformaszigbee.blogspot.com.es/2012/05/practica-1-configuracion-y-conceptos.html.

12. **Muguira, Leire, y otros.** DeustoTech. [En línea] 2007. [Citado el: 23 de Agosto de 2017.]

http://www.tecnologico.deusto.es/projects/smartmotes/files/D2.1_SmartMotes_WSN_ComparativeAnalysis_v1.8.pdf.

13. **nettigo. [En línea] [Citado el: 24 de Agosto de 2017.] https://nettigo.eu/products/wifi-module-nodemcu-v3-lolin-with-esp-12f.**

14. Lextrait, Thomas. THOMAS LEXTRAIT. [En línea] 22 de Mayo de 2016. [Citado el: 24 de Agosto de 2017.] https://tlextrait.svbtle.com/arduino-power-consumption-compared.

15. Geerling, Jeff. Jeff Geerling. [En línea] 27 de Noviembre de 2015. [Citado el: 24 de Agosto de 2017.] https://www.jeffgeerling.com/blogs/jeff-geerling/raspberry-pi-zero-power.

16. Aranda, Jorge. http://pensamiento-logico.rhcloud.com. [En línea] 24 de Julio de 2016. [Citado el: 29 de Agosto de 2017.] http://pensamiento-logico.rhcloud.com/esp-12-vs-arduino-uno-r3-vs-arduino-nano/.

17. *A comparative study of wireless protocols: Bluetooth, UWB, ZigBee, and Wi-Fi.* Lee, Jin-Shyan, Su, Yu-Wei y Shen, Chung-Chou. 2007. Industrial Electronics Society, 2007. IECON 2007. 33rd Annual Conference of the IEEE. págs. 46-51.

18. Espressif Inc. http://espressif.com/. *http://espressif.com/.* [En línea] Abril de 2017. [Citado el: 29 de Agosto de 2017.] http://espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf.

19. http://lwip.wikia.com. [En línea] [Citado el: 2017 de Agosto de 29.] http://lwip.wikia.com/wiki/LwIP_Wiki.

20. Ivan, Grokhotkov. https://arduino-esp8266.readthedocs.io. *https://arduino-esp8266.readthedocs.io/en/latest/reference.html.* [En línea] 2017. [Citado el: 4 de Septiembre de 2017.] https://arduino-esp8266.readthedocs.io/en/latest/_images/esp12.png.

21. Espressif Inc. https://www.espressif.com. *https://www.espressif.com.* [En línea] 2016. [Citado el: 4 de Septiembre de 2017.]

https://www.espressif.com/sites/default/files/9b-esp8266-low_power_solutions_en_0.pdf.

22. https://www.allaboutcircuits.com. [En línea] 23 de Junio de 2016. [Citado el: 1 de Septiembre de 2017.] https://www.allaboutcircuits.com/technical-articles/how-why-of-energy-harvesting-for-low-power-applications/.

23. *Energy conservation in wireless sensor networks: A survey.* Anastasi, Giuseppe, y otros. s.l. : Elsevier, 2009, Ad hoc networks, Vol. 7, págs. 537-568.

24. *Survey on wakeup scheduling for environmentally-powered wireless sensor networks.* Valera, Alvin C., Soh, Wee-Seng y Tan, Hwee-Pink. s.l. : Elsevier, 2014, Computer Communications, Vol. 52, págs. 21-36.

25. Capella Hernández, Juan Vicente. *Redes inalámbricas de sensores:Una nueva arquitectura eficiente y robusta basada en jerarquía dinámica de grupos.* Valencia : Universitat Politècnica de València , 2011.

26. Garrido Campo, Gonzalo. Universidad de Málaga. [En línea] Junio de 2009. [Citado el: 5 de Septiembre de 2017.] http://webpersonal.uma.es/~ECASILARI/Docencia/Memorias_Presentaciones_PFC/49_Memoria_GonzaloCampos.pdf.