Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Diseño e implementación de una red inalámbrica de sensores para la detección precoz de incendios forestales

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Rubén Quintanilla Lahiguera

Tutor: Alberto Miguel Bonastre Pina

2018-2019

Combatir el fuego con fuego…

A mis padres… por el gran apoyo que me han dado durante años para realizar esta carrera. Sólo ellos saben lo mucho que significaba para mí, pero no lo mucho que ellos significan para mí.

Agradecimientos

Quiero agradecer a Alberto Bonastre Pina, mi tutor en este proyecto, aguantar mis idas y venidas durante la realización de este proyecto, las tantas veces que me ha apoyado y animado a continuar cuando yo creía encontrar el componente que necesitaba, pero me acababa desilusionando porque no me servía. Gracias por compartir tu ilusión y tus conocimientos. Aunque han sido pocas las reuniones, han sido de mucha ayuda. De verdad, gracias.

Agradecer también a Miguel A. Sislian Suez (que más que un socio, es un buen amigo), el escuchar todas mis paranoias y problemas surgidos en este proyecto y apoyarme para superar todas las adversidades que surgían conforme avanzaba. Gracias también por no enfadarte cuando no te contesto a los mensajes durante días. Tienes grandes ideas.

Y por supuesto quiero dar las gracias a Nathan Heatley por aguantarme en esos días en los que he estado muy estresado. Gracias también por escuchar mis explicaciones sobre ideas, teorías y algoritmos de este proyecto, aunque sé que sólo te enterabas de la mitad. Gracias por preocuparte y estar ahí cuando me quedaba atascado. Tu apoyo ha sido una gran ayuda.

Resumen

El objetivo de este proyecto es diseñar un sistema de detección de incendios forestales basado en una red inalámbrica de sensores. Para llevarlo a cabo, se realiza un estudio de las redes de sensores, analizando sus características, estructura, topologías de red, algoritmos de enrutamiento y opciones de conectividad inalámbrica de largo alcance más comunes. También se hace un breve repaso a la problemática que existe sobre los incendios forestales y la utilidad que IoT y las Smart Cities pueden ofrecer para la prevención de incendios forestales.

Una vez comprendida la naturaleza de las redes de sensores y el papel que pueden jugar en la detección de incendios forestales, se plantea una solución hardware basada en la creación de nodos a medida. También se desarrolla el algoritmo que ejecutan los nodos, el cual implementa una red en malla junto con un enrutamiento por inundación. A continuación, se construye un prototipo en Arduino que implementa el algoritmo desarrollado para, así, comprobar si el objetivo es alcanzable. Para ello, se diseña un plan de pruebas de cuatro partes que comprueban las funcionalidades más importantes, a la vez que evalúan la tolerancia a fallos, la adaptabilidad, la escalabilidad, la correctitud y la fiabilidad e integridad del sistema. Una vez realizadas las pruebas, se obtienen resultados muy prometedores. Lo cual demuestra que, haciendo uso de las redes inalámbricas, la topología en malla y el enrutamiento por inundación, se ha desarrollado un sistema de detección de incendios forestales viable y listo para su implantación.

Palabras clave: red inalámbrica de sensores, bluetooth, bajo consumo, largo alcance, incendios forestales, iot, smart city

Resum

L’objectiu d’aquest projecte és dissenyar un sistma de detecció d’incendis forestals basat en una xarca de sensors sense fil. Per a dur-ho a terme, es realitza un estudi de les xarxes de sensors, analitzant les seues característiques, estructura, topologies de xarxa, algoritmes de encaminament y opcions de connectivitat sense fil de llarg abast més comuns, També es fa un breu repàs a la problemàtica que existeix sobre el incendis forestals i la utilitat que IoT i les Smart Cities poden oferir par a la prevenció d’incendis forestals.

Una vegada compresa la naturalesa de les xarxes de sensors i el paper que poden jugar en la detecció d’incendis forestals, es planteja una solució hardware basada en la creació de nodes a mesura. També es desenvolupa l’algorisme que executen els nodes, el qual implementa una xarxa en malla juntament amb un encaminament per inundació. A continuació, es construeix un prototip en Arduino que implementa l'algorisme desenvolupat per a, així, comprovar si l'objectiu és assolible. Per a això, es dissenya un pla de proves de quatre parts que comproven les funcionalitats més importants, alhora que avaluen la tolerància a fallades, l'adaptabilitat, l'escalabilitat, la correció i la fiabilitat i integritat del sistema. Una vegada realitzades les proves, s'obtenen resultats molt prometedors. La qual cosa demostra que, fent ús de les xarxes sense fils, la topologia en malla i el encaminament per inundació, s'ha desenvolupat un sistema de detecció d'incendis forestals viable i preparat per a la seua implantació.

Paraules clau: xarxes de sensors sense fils, bluetooth, baix consum, llarg abast, incendis forestals, iot, smart city

Abstract

The objective of this project is to design a forest fire detection system based on a wireless sensor network. To carry this out, a study of wireless sensor networks is conducted, which analyses their characteristics and structure and the most common network topologies, routing algorithms and options for long-range wireless connectivity. A brief review of the issue of forest fires and the utility that the IoT and smart cities have to offer towards the prevention of forest fires is also done.

Once the nature of sensor networks and the role they play in the detection of forest fires are understood, a hardware solution, based in the creation of tailor-made nodes, is proposed. The algorithm which the nodes use is also developed, which implements a mesh network together with a flooding routing algorithm. As a continuation, a prototype is constructed in Arduino which implements the developed algorithm in order to test if the objective is obtainable. To this end, a test plan of four parts is designed, which checks the most important functionalities and at the same time, demonstrates the tolerance to failures, adaptability, scalability and correctness, reliability and integrity of the system. Once the tests are conducted, very promising results are obtained. This demonstrates that (whilst making use of wireless networks, mesh topology and flooding routing algorithms) a viable system for the detection of forest fires has been developed, which is ready for implementation.

Keywords: wireless sensor network, bluetooth, low energy, long range, forest fire, iot, smart city

Tabla de contenidos

Tabla de imágenes

1. Introducción

En este apartado se hará una pequeña introducción para poder situarse en el contexto que plantea el proyecto. Para obtener una mejor visión del planteamiento, se definirá la misión y los objetivos de este, sin olvidar de situar el ámbito de aplicación. Adicionalmente se presentará las motivaciones que han llevado a plantear y realizar este proyecto.

1.1. Contexto

En la actualidad, las redes inalámbricas se distribuyen a lo largo del planeta de diversas formas, permitiendo comunicarnos de formas diversas según nuestras necesidades. Tales como Wi-Fi y Bluetooth pueblan nuestras ciudades, mientras que las redes de telefonía móvil y radiodifusión se expanden por gran parte del planeta, dando servicio a millones de personas a la vez. La comunicación inalámbrica cambió nuestras vidas y, hoy en día siguen evolucionando, mejorando sus prestaciones y capacidades.

Si bien parece que la tecnología y el medio ambiente pueden ser contrarios entre sí, podemos utilizar la tecnología para solventar problemas medioambientales o por lo menos atenuarlos. Gracias al uso de las energías renovables, podemos obtener pequeñas fuentes de energía casi en cualquier lugar. Junto con la evolución que han tenido las baterías en estos últimos años y la reducción del consumo de los dispositivos de última generación, podemos introducir la tecnología casi donde la necesitemos.

Es por ello por lo que podemos plantearnos integrar las comunicaciones inalámbricas en nuestro ecosistema para prevenir ciertas catástrofes que, en cierta medida, están fuera del alcance de las personas.

Una de las catástrofes más escuchadas en nuestro país han sido los incendios forestales ocurridos en nuestro país durante los años 2012 y 2017. Los incendios forestales que se producen cada año arrasan con miles de hectáreas, empobreciendo el ecosistema de nuestro país y afectando a la flora y fauna de las regiones [7].

Mediante este proyecto, se pretende reducir, de una forma económica y poco invasiva para el medio ambiente, la deforestación resultante de los incendios forestales. Problema que nos afecta a todos y que diezma la población de bosques en todo el mundo.

1.2. Motivaciones

La motivación para realizar este proyecto viene de hace varios años atrás, cuando comencé la carrera. Ya era un proyecto que tenía en mente cuando finalicé mis estudios de FP Grado Superior, pero no poseía los conocimientos necesarios para su realización.

Siempre he estado interesado en la informática, pero también en la ecología, por lo que la unión de ambos conceptos me parece muy interesante, con muchas perspectivas de futuro y con una gran aplicación, tanto en el presente como en el futuro cercano. Dado la aparición y crecimiento de las Smart Cities, hace que me resulte más interesante cómo este tipo de proyectos puede cambiar la sociedad.

En los últimos años han ocurrido muchos incendios en nuestra comunidad autónoma y en el resto del país. De forma pasiva, observamos en las noticias cada año los desastres que mellan en nuestros bosques, casi cotidiano, restándole importancia. Este hecho refuerza en mí la realización de este proyecto como Trabajo de Fin de Grado, además de ser todo un reto para culminar mis estudios. Sin duda, un proyecto que me gustaría completar y hacerlo realidad.

1.3. Misión del proyecto

Diseñar una red de sensores para la detección precoz de incendios forestales.

1.4. Ámbito de aplicación

Este proyecto puede aplicarse tanto en pequeñas parcelas arboladas, como en grandes extensiones de bosques. Principalmente, está pensado para cubrir las parcelas con un mayor riesgo de incendio, mayor impacto medioambiental o reservas protegidas.

Puede ser aplicado tanto para el ámbito público como privado, ya que se presenta un sistema de bajo coste. Al ser un sistema altamente escalable.

1.5. Objetivos

El objetivo principal de este proyecto es diseñar una red de sensores capaz de detectar posibles incendios en zonas arboladas, e implementar una interfaz gráfica en

un servidor donde poder mostrar los datos obtenidos. Para ello, se debe cumplir unos objetivos específicos que constarían de:

- Comparar las topologías, tecnologías de conectividad y algoritmos de enrutamiento más usuales de una red
- Determinar las funciones de cada integrante de la red de sensores
- Analizar el problema de los incendios forestales y obtener una solución al problema planteado
- Analizar brevemente la situación actual del mercado sobre dispositivos sensores
- Determinar los componentes adecuados que deberán tener los dispositivos sensores según las especificaciones
- Diseñar una solución al problema planteado mediante el desarrollo de una red de sensores
- Implementar un prototipo de la solución propuesta
- Comprobar que la solución planteada funciona

2. Estado del arte

En este apartado se explicará la importancia que han tenido las redes de sensores en la historia y la importancia que tienen hoy en día para IoT y las Smart Cities.

A continuación, se realizará una descripción de sus características, funcionamiento y topologías de red que pueden adoptar, abordando sus ventajas e inconvenientes. También se hará una pequeña descripción de algunas tecnologías de conectividad inalámbricas de largo alcance que existen en la actualidad, y se analizarán y describirán algunos dispositivos disponibles a la venta en el mercado.

Finalmente, se hará una breve introducción sobre la importancia de los incendios forestales mediante datos estadísticos. Se describirán sus tipologías, causas y se enumerarán los elementos generados en la combustión. De esta forma, se enlazará las posibilidades que ofrecen las Smart Cities frente al problema de los incendios forestales.

2.1. Historia de las redes de sensores

Un hecho histórico de las redes de sensores es que su origen es algo difuso y oculto. Esto es debido a que, originalmente, se trataba de una iniciativa militar y es por ello por lo que se carece de información al respecto.

Uno de los primeros proyectos que se conocen, catalogado como red de sensores, es el Sound Surveillance System[1] (SOSUS). Este proyecto fue creado por la Armada de Estados Unidos durante la Guerra Fría en el año 1950. Consistió en una red de boyas sumergidas en el agua, capaces de detectar la presencia de submarinos en sus alrededores mediante los sensores acústicos que incorporaban, llamados hidrófonos [1].

Otro proyecto militar también interesante fue Igloo White, desarrollado más tarde en 1960 durante la guerra de Vietnam. Dicho proyecto consistió en una red de sensores capaz de dar la alarma al detectar cuando el enemigo cruzaba la zona desmilitarizada. Mediante el uso de sensores sísmicos y acústicos, eran capaces de detectar incluso pisadas y explosiones [2].

Este tipo de proyectos despertaron la curiosidad de científicos y organizaciones. Lo cual desembocó que, en 1980, se iniciase la investigación en redes de sensores mediante el proyecto Distributed Sensor Networks, de la Defense Advanced Research Projects Agency (DARPA), de Estados Unidos.

Las redes de sensores siguen evolucionando hoy en día y están más presentes que nunca. Algunos proyectos interesantes en la actualidad son:

UbiMon: es un proyecto actual del ámbito de la medicina. Se trata de una serie de biosensores capaces de medir datos biométricos de pacientes y realizar el seguimiento de estos mediante el uso de una red inalámbrica de sensores. Su uso está destinado principalmente a pacientes recientemente operados que requieren cierto control [3].

SNOTEL[2]: se trata de una red inalámbrica de sensores creada a mediados de los años sesenta, situada al oeste de Estados Unidos y utilizado por el Servicio de Conservación de Recursos Naturales de este mismo país. Consiste en una serie de estaciones capaces de medir una gran cantidad de datos climatológicos para poder predecir, no sólo el clima, sino también inundaciones o incluso pronosticar el suministro anual del agua. Se comunican mediante señales de muy alta frecuencia (Very High Frequency o VHF) y particularmente aprovechan la tecnología de comunicación por ráfagas de meteoritos (MBC) para reflejar la señal y así obtener datos casi en tiempo real.

Desde entonces hasta el día de hoy, las redes de sensores han ido ganando importancia en la evolución de la tecnología y en la mejora de la vida de las personas.

[1] https://www.globalsecurity.org/intell/systems/sosus.htm#1990
[2] https://www.wcc.nrcs.usda.gov/snow/

2.2. IoT y las Smart Cities

Durante los últimos años, las redes de sensores han ido evolucionando rápidamente y han acabado integrándose en nuestras vidas sin darnos cuenta. A pesar de que comenzó como un proyecto militar, se ha ido abriendo paso a otros sectores como la medicina, el medio ambiente, ahorro energético o incluso el transporte, entre otros. En todos ellos, su presencia es muy importante, ya que facilita la obtención de datos y el acceso a éstos.

Perteneciente a las siglas Internet of Things, IoT es un término surgido al otorgar a los dispositivos y electrodomésticos cotidianos, acceso a internet y todas sus ventajas. Es decir, mediante la interconexión de distintos dispositivos, podemos dotarles de capacidad para realizar ciertas tareas simples y compartir información entre ellos. Esto mismo les permite realizar tareas cooperativas entre ellos para facilitar la vida de las personas. Además, dando conectividad a Internet a estos dispositivos, ampliamos sus funcionalidades y el alcance de estas. Un claro ejemplo son las neveras inteligentes, capaces de realizar la compra casi de forma autónoma. Otro ejemplo sería las lavadoras o calefacción inteligente, capaces de configurarse a distancia. No sólo se encuentran ejemplos actuales en electrodomésticos, son muy comunes los dispositivos como las smartbands, smartphones, smartwatches o incluso los vehículos.

Si bien IoT es aplicado en el ámbito personal, también puede aplicarse en al ámbito social a una mayor escala. A esto se le llama Smart Cities, o ciudades inteligentes o digitales. El concepto surge de aplicar IoT en el núcleo urbano. Mediante el uso de sensores muy diversos se pueden obtener datos de gran utilidad para la gestión eficiente de la ciudad. Los sensores están repartidos por la ciudad y todos los datos son recogidos, procesados y servidos a la comunidad mediante plataformas accesibles a cualquier persona. La información resultante de estos datos es de gran interés científico y estadístico, pero más aún para la ciudadanía y el aprovechamiento de los recursos. El conjunto de estos beneficios se puede identificar y categorizar en 6 dimensiones o medidas: economía, movilidad, medio ambiente, ciudadanía, vida y gobierno [5] (Ilustracion1). Algunos ejemplos beneficiosos para las ciudades son los localizadores de transporte público, los indicadores de vaciado de contenedores, los medidores de calidad del aire, los medidores de volumen de tráfico, estado de los jardines, ocupación de plazas de aparcamiento, etc [4].

Ilustración 1. Ventajas de las Smart Cities

En España podemos encontrar el proyecto SmartSantander[3]. Es un proyecto pionero de smart city implementado en la ciudad de Santander desde 2009. Este proyecto cuenta con más de 20000 sensores repartidos por la ciudad, los cuales miden factores ambientales como temperatura, humedad, viento, calidad de aire, etc. También están incorporados en medios de transporte como autobuses y taxis, por lo que también miden velocidad y posición. Dado que toda esa información abierta y accesible, se pone a disposición del ciudadano mediante diversas plataformas [4].

Sin duda, las Smart Cities son el futuro de las ciudades. Suponen una gran ventaja, tanto económica como social, que mejora la vida de los ciudadanos que viven en ellas entre un 10% y un 30% en muchas de las medidas [5].

2.3. Características de las redes de sensores

Una red de sensores, o Distributed Sensor Network por sus siglas DSN, se trata de un gran conjunto de sensores inteligentes, generalmente idénticos o con pequeñas diferencias, interconectados y distribuidos geográficamente en un entorno concreto. Estos sensores pueden estar especializados en una gran variedad de características a medir de su entorno. Tal es así que podemos encontrar desde los más simples como sensores de temperatura, humedad y presión; hasta los más complejos como cámaras, sensores de radiactividad o sensores sísmicos [4].

Los sensores toman los datos de su entorno de una forma continua, ya sea de forma ininterrumpida o durante unos intervalos de tiempo específicos definidos por

[3] http://www.smartsantander.eu/

densidad de datos o por necesidades energéticas. Una vez los datos han sido tomados y procesados, son enviados a través de la red a la que están conectados.

Los datos circulan entre la red transmitiéndose entre nodos de forma lógica y coherente. Dependiendo del tipo de red, pueden ser necesarios nodos de enrutamiento. Para el reenvío de datos se suele usar alguna estrategia de fusión de datos que permite reducir, de forma considerable, la cantidad de datos que circulan por la red y así evitar su congestionamiento.

Dependiendo del tipo de necesidades, los datos recogidos por cada sensor pueden ser distribuidos a todos los sensores restantes de la red, permitiendo conocer el estado de todos ellos desde cualquier punto de la red de sensores. Otra necesidad podría ser que los propios sensores se encarguen de redirigir, o enrutar, los datos recogidos por cada sensor hacia uno o varios extremos de la red, llamados sumideros.

Finalmente, los datos son transmitidos a un dispositivo sumidero, capaz de procesar la información recogida e integrada por los sensores de la red y hacer uso de ellos para cumplir el objetivo por el cual fueron tomados.

Una variante que podemos encontrar de este tipo de red son las Wireless Sensor Network, de las siglas WSN. Una WSN consiste en la unión de las redes de sensores con la tecnología inalámbrica, De esta forma, se puede aprovechar todas las ventajas que ofrecen las conexiones inalámbricas a las DSN, pudiendo reducir los costes, aumentar el alcance y dotar de movilidad. Esto aumenta la comodidad a la hora de implementar DSNs, por lo que actualmente es una ventaja muy competitiva a la hora de plantear el diseño de una DSN.

2.4. Topologías de red en las WSN

Las topologías de red que podemos encontrar hoy en día son muy específicas y concretas. Aunque son limitadas, son suficientes para abarcar cualquier necesidad. A continuación, analizaré las características de las topologías de red más acordes para cumplir las especificaciones:

2.4.1. Topología de red en estrella

Este tipo de topología tiene un nodo central, al que todos los demás nodos están única y exclusivamente conectados (Ilustración 2). El nodo raíz canaliza y coordina las conexiones del resto de nodos, controlando todas las transmisiones de datos. En caso de que un nodo quiera comunicarse con otro, necesariamente tendrá que hacerlo a través del nodo central. Es decir, la información se transmite directamente al nodo central y éste a los demás.

Como ventaja, podemos observar que el hecho de añadir o quitar nodos a la red, no afecta a la misma, al igual que cuando un nodo falla, sólo afectará a su propia comunicación, dejando el resto de la red intacta. Por contra, siendo el nodo raíz el que controla todas las transmisiones, concentrará toda la carga de red, pudiendo afectar al estado de esta. Por la misma razón, si éste sufriera una caída, la red entera dejaría de funcionar.

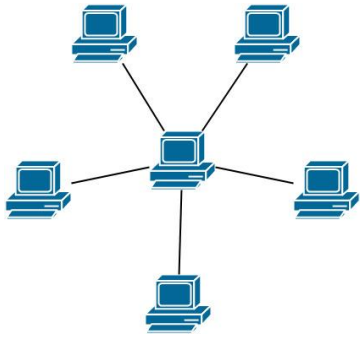

Ilustración 2. Topología en estrella

2.4.2. Topología de red en árbol

Este tipo de topología tiene un nodo raíz el cual tiene nodos que dependen de otros nodos a su vez, de forma jerárquica y de como si de la estructura de un árbol se tratase (Ilustración 3). De esta forma, la información se va transmitiendo de un nodo a su superior inmediato hasta llegar al nodo raíz. Esta topología puede verse como un conjunto de topologías en estrella anidadas, por lo que su funcionamiento y características son muy similares.

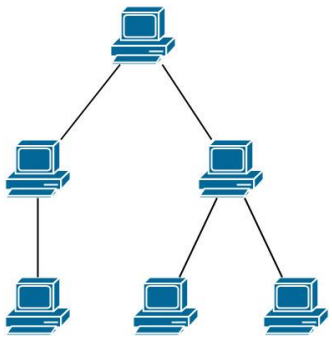

Ilustración 3. Topología en árbol

2.4.3. Topología de red en anillo

En este tipo de topología, los nodos se conectan entre ellos en serie y el último con el primero, formando un círculo o anillo (Ilustración 4). De esta forma, los datos circulan por todos los nodos del camino hasta llegar al nodo destino. Para comunicar un nodo con otro, tienen que reservar el canal de comunicación hasta que finaliza la transmisión, entonces lo deja libre para que otro nodo pueda utilizarlo. Esto hace que

no se requiera de un nodo central que administre las comunicaciones en la red, por lo que es un ahorro en infraestructuras. Para ello, los propios nodos se encargan de administrar la red y el efecto del fallo de uno de ellos pone en riesgo la estructura y las comunicaciones de los demás nodos de la red, ya que los datos no pueden continuar por el camino que deberían. Por otro lado, hay que tener en cuenta que, al ser nodos interconectados unos de otros, la transmisión será más rápida si se trata de un nodo vecino, además de hacer más sencilla la localización del fallo en la red para su posterior solución, puesto que se puede realizar un seguimiento de los datos enviados.

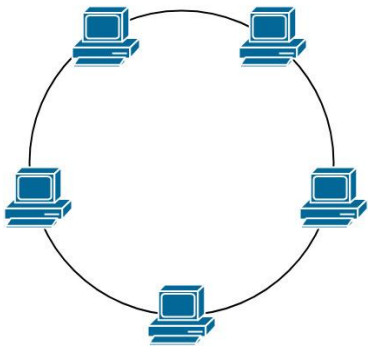

Ilustración 4. Topología en anillo

2.4.4. Topología de red en malla o mesh

En este tipo de topología, los nodos se conectan entre sí sin ningún orden jerárquico (Ilustración 5). Esto hace que existan diversos caminos para transmitir los datos hasta el destino y, si un nodo falla, puede derivarse la información a otro nodo y continuar el camino por una ruta alternativa sin afectar a la red. Esto mismo puede ser útil cuando una parte de la red está congestionada o caída. Detectando dicha eventualidad, se puede escoger un camino alternativo válido y así evitar retrasos en la circulación de los datos por la red. A su vez, esto puede ser un problema, ya que puede ser difícil identificar fallos en la red, al poder confundirlos con rutas alternativas o, directamente, no llegar a reconocerlos. El mayor desafío en esta topología es el enrutamiento de los datos, para lo cual existen diversas técnicas. Es la forma más utilizada hoy en día, pues al aumentar el número de nodos, aumenta la tolerancia a fallos. Aunque, por el contrario, la hace más difícil de administrar.

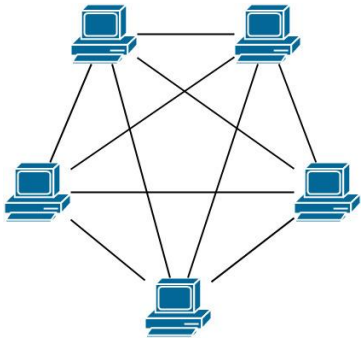

Ilustración 5. Topología en malla

2.5. Algoritmos de enrutamiento de datos en redes

En la actualidad existen muchos algoritmos de enrutamiento de datos, algunos de ellos más utilizados que otros, pero siempre dependen de las necesidades de cada red. Algunos ejemplos son el enrutamiento por ruta más corta, vector distancia, estado del enlace, jerárquico, inundación, difusión, etc [6]. Los algoritmos de enrutamiento se pueden clasificar en estáticos o dinámicos. Los algoritmos de enrutamiento estáticos se caracterizan por no cambiar sus rutas prácticamente, a no ser que se hagan manualmente. Por otro lado, los algoritmos de enrutamiento dinámico evolucionan o cambian sus rutas de forma continua, dependiendo de varios factores de la red. A continuación, se va a explicar las características de algunos de ellos:

2.5.1. Vector distancia

Clasificado como un algoritmo de enrutamiento de tipo dinámico, consiste en el uso de una ecuación, llamada ecuación de Bellman-Ford, para que cada enrutador genere una tabla con el cálculo de la distancia o saltos para llegar a cada destino de la red por cada uno de sus enlaces. De esta forma, compartiendo sus tablas a todos los nodos y cruzando sus datos, llegan a saber qué enlace tiene el camino mínimo hasta el destino. Es decir, tiene en cuenta la distancia hasta el destino, sin importar la saturación de los enrutadores que intervienen. RIP, BGP y IGRP son algunos protocolos de enrutamiento que utilizan este algoritmo [6].

2.5.2. Estado de enlace

Clasificado como un algoritmo de enrutamiento de tipo dinámico, consiste en que cada enrutador tiene que descubrir los vecinos de su alrededor y el coste para llegar a cada uno de ellos. Esta información es enviada a sus nodos vecinos que procesan los datos mediante el algoritmo de Dijkstra para calcular el camino más corto hasta el destino y vuelven a compartir los datos obtenidos, creándose un mapa virtual de la red en cada enrutador. De esta forma, este algoritmo tiene en cuenta el estado, tanto de los enrutadores, como de los enlaces, pudiendo buscar caminos alternativos en casos de congestión de la red. Se trata de un algoritmo de enrutamiento muy usado hoy en día. OSPF e IS-IS son unos de los protocolos que lo utilizan [6].

2.5.3. Inundación controlada

Clasificado como un algoritmo de enrutamiento estático ya que carece de adaptabilidad o cambios. Consiste en que cada enrutador reenvía los datos por todos sus enlaces menos por el que le llegó, sin importar el destino ni la ruta a seguir. Para evitar ciclos en la inundación, se comprueba si el paquete ya ha pasado por dicho enrutador comprobando el número de secuencia.

Realmente es una forma de enrutamiento basta pero simple, que se podría llamar de “fuerza bruta”. Una ventaja, o inconveniente en términos de seguridad, reseñable de este algoritmo de enrutamiento, es que los datos enviados se distribuirán por toda la red. A pesar del inconveniente de que puede llegar a saturar la red, es un algoritmo efectivo [6].

2.6. Tecnologías de conectividad inalámbricas de largo alcance

Analizando el mercado, podemos encontrar las siguientes tecnologías de conectividad inalámbrica que adoptan protocolos o funcionalidades de bajo consumo, y que más se ajustan a las especificaciones de este proyecto (Tabla 1).

Technologia	Frecuencias/ GHz	Velocidad	Alcance
Bluetooth 5 LR	2,4 - Sub-GHz	1 Mbps	500m
LoRaWAN	Varias	0.3 - 50 kbps	5km
SigFox	Sub-GHz	0.3 kbps	30km
Wi-Fi ah	2,4 - 5 - Sub-GHz	150 kbps- 346 Mbps	1km (900 MHz)
Zigbee	2,4	250 kbps	10- 100m

Tabla 1. Comparativa de Tecnologías de conectividad inalámbrica

2.6.1. Bluetooth Long Range[4]

Bluetooth Long Range, o BLR, como su nombre indica es la versión de largo alcance, además de bajo consumo, de Bluetooth. Capaz de competir con los tradicionales SigFox y LoRa. Actualmente también coexiste con Bluetooth Low Energy, BLE, que se trata de la versión de bajo consumo y corto alcance de Bluetooth. Como ventaja más competitiva, la tecnología Bluetooth cuenta con una estandarización y despliegue a nivel global en la mayoría de los dispositivos comercializados actualmente. Desde la versión 4.0 se han implementado características de redes mesh o malla que pueden integrase. Esta característica le hace tener una gran ventaja, ya que le permite interconectar los dispositivos unos con otros, sin necesidad de enrutadores, y así ampliar su alcance.

Inicialmente bluetooth utilizaba topología en estrella. Para que suportase redes de malla se implementaron diferentes soluciones, entre ellas están BLE Mesh Network

[4] https://www.bluetooth.com/

y CSRmesh. Las diferencias entre éstas, a grandes rasgos, es que BLE Mesh Network utiliza un enrutamiento dinámico, el cual calcula la ruta óptima hasta el destino. En grandes redes esto puede ser un problema, puesto que se puede congestionar el nodo raíz. Por otro lado, CSRmesh propone una solución al enrutamiento basado en difusión o inundación. Los datos se difunden por todos los nodos de la red, descartándolos cuando detectan que son duplicados. Adicionalmente, permite el envío a varios destinatarios. Esto también supone una gran ventaja ya que, al ser toda la red conocedora de los datos, pueden situarse varios nodos sumidero en distintos puntos de la red. [6]

2.6.2. LoRaWAN[5]

Proviene de la abreviatura de Long Range Wide Area Network, que significa red de área de largo alcance. Se trata de una especificación para redes de dispositivos de bajo consumo y largo alcance y su arquitectura de red típica es una red de redes en estrella. Además, está diseñado para poder contactar millones de dispositivos, por lo que es usado para IoT. LoRaWAN está basado en el espectro ensanchado, (técnica que con anterioridad fue utilizada para la comunicación en el espacio y por el ejército) por lo que las comunicaciones de datos no se ven afectadas por otras a velocidades distintas. Opera en la frecuencia de los 868 MHz en Europa y es capaz de transmitir datos hasta 50 kbps con un alcance de 10 a 20 Km, dependiendo de las inclemencias climatológicas y del entorno.

2.6.3. SigFox[6]

Es una de las más conocidas ya que usa la banda ultra estrecha, que le confiere capacidad para atravesar objetos sólidos y poder dar cobertura incluso a objetos subterráneos. Trabaja en la frecuencia del Sub-1GHz y tiene un alcance de entre 3 y 50 km dependiendo del entorno y opera bajo una topología en estrella de un salto.

SigFox es una compañía francesa fundada en 2009 en busca de convertirse en un proveedor de redes IoT. A diferencia de muchos de sus competidores, posee una gran infraestructura que le otorga una amplia cobertura en un conjunto de 50 países,

[5] https://lora-alliance.org/
[6] https://www.sigfox.com/en

con un total de 4,2 millones de kilómetros cuadrados de cobertura. Esto mismo hace que sea muy práctico para el usuario utilizar esta red, pues se despreocupa del estado de la red y la configuración de esta. Al ser una red dependiente del propietario, el usuario debe pagar una suscripción para hacer uso de la misma por cada dispositivo que se conecta, De esta forma, la compañía SigFox asigna una limitación en la conexión de dispositivos y otra limitación de 140 mensajes diarios a la transmisión de datos.

2.6.1. Wi-Fi ah o HaLow[7]

De la abreviatura Wireless Fidelity. Esta versión de Wi-Fi fue estandarizada en marzo de 2016 y está optimizado para conectar hasta 8191 dispositivos sin perjudicar a la red. Es por esto por lo que se puede decir que ha sido diseñado especialmente para su uso en IoT. Trabaja en la frecuencia del sub-1 GHz, lo que le confiere un mayor alcance de hasta 1 Km y una velocidad de transmisión de hasta 100Kbps. Dado su principal uso, fue desarrollado para ahorrar energía, por lo que tiene un consumo bajo. Usualmente sigue una topología en estrella, tal como sus antecesores versiones de Wi-Fi.

2.6.2. ZigBee[8]

Se trata de una especificación de un conjunto de protocolos para la transmisión de datos de forma inalámbrica y con un consumo muy reducido, lo que lo hace ideal para dispositivos con una alimentación eléctrica limitada. Se basa en el estándar IEEE 802.15.4 y opera con las frecuencias de 868 MHz, 915 MHz y 2.4 GHz. Muy parecido a Bluetooth, pero de menores prestaciones. Actualmente está ganando mucha importancia en el mundo IoT y la domótica, así como en la industria y en la medicina.

2.7. Funcionamiento de los motes

Los nodos sensores, o también llamados motes, son pequeños dispositivos de bajo consumo, capaces de obtener datos de su entorno para poder transmitirlos a través de la red de sensores. Es por ello, que todo nodo que es partícipe de una red de sensores debe cumplir una arquitectura hardware concreta que desarrollará sus

[7] https://www.wi-fi.org/discover-wi-fi/wi-fi-halow
[8] https://www.zigbee.org/

funciones como miembro de la red de sensores. Las unidades funcionales básicas con las que deben contar son: Sensor, Unidad Computacional y Radiotransmisor.

Esto se puede explicar mediante una ejecución básica del proceso de funcionamiento de uno de los motes:

La ejecución comienza con los sensores, los cuales tomarán los datos necesarios y serán transferidos a la unidad computacional. La unidad Computacional es usualmente un microcontrolador que procesa los datos de los sensores, los transforma y realiza con ellos las operaciones pertinentes. Una vez la unidad computacional ha realizado su tarea, envía los datos resultantes al radiotransmisor, que es un dispositivo de comunicaciones inalámbricas capaz de transferir los datos a sus nodos vecinos mediante algún protocolo específico.

Todo esto quiere decir que, para el diseño de los motes habrá que tener en cuenta la arquitectura anteriormente descrita, es decir, qué sensores utilizar para obtener los datos de interés del entorno, qué microcontrolador utilizar respetando y siguiendo las especificaciones descritas, y qué radiotransmisor utilizar que se ajuste también a las especificaciones.

2.8. Motes comerciales

En el mercado, hay empresas que desarrollan este tipo de motes de forma genérica. Estos motes suelen ser bastante simples y con pocas funciones, pero tienen la posibilidad de ampliar sus características mediante la inserción de componentes o ampliaciones.

A continuación, se presentan algunos ejemplos. Todos ellos tienen conectividad inalámbrica, sensores de temperatura y humedad integrados y, a excepción de uno de ellos, con soporte de topología de malla:

panStamp NRG3[9]*:* Creado por la empresa panStamp. Se trata de un mote que posee un microcontrolador CC430F5137, con una velocidad de 8 – 24 MHz. La tensión de funcionamiento es de entre 2 y 3,6 V y en modo Sleep consume 2,5 µA. Tiene un alcance de entre 1 y 5 Km, configurable dependiendo del modo seleccionado. El precio por cada nodo es 36€ (Ilustración 6).

[9] http://www.panstamp.com/product/panstamp-nrg3/

Ilustración 6. Mote panStamp NRG3

WiSense WHTN155[10]*:* Creado por la empresa WiSense Technologies. Posee varios modelos de motes, unos dedicados y otros genéricos a los que se les puede integrar diversos sensores. Posee un microcontrolador CC1120 capaz de variar su velocidad de entre 8 y 16 MHz. La tensión de funcionamiento es entre 1,8 y 3,6 V, con un consumo de 1 µA en modo Sleep. Tiene un alcance de hasta 1 Km y su precio es alrededor de los 18 € (Ilustración 7).

Ilustración 7. Mote WiSense Sensor Node

[10] http://wisense.in/

BPart[11]*:* La empresa TECO nos presenta este mote con conectividad Bluetooth Low Energy. Tiene un chip BLE112 que usa, tanto como unidad computacional, como radiotransmisor. La desventaja de este chip es que no puede adoptar topología de malla. Alcanza una velocidad de 32 MHz. Funciona con una tensión de 3 V, consume 20 µA y su alcance es de 400 m. Como ventaja, incluye sensores para numerosos gases diferentes. Su precio es de 40€ (Ilustración 8).

Ilustración 8. Mote BPart

Tmote Sky MTM-CM5000-MSP[12]*:* Es uno de los motes creados por la empresa Telosb Sensors. Posee un microcontrolador MSP430F1611 a una velocidad de 8 MHz. Funciona a una tensión de 3 V con un consumo de 1 µA. Como peculiaridad, funciona también con redes ZigBee, pero con un alcance limitado de 120 m. Su precio en el mercado es alrededor de 90 € según el modelo. (Ilustración 9)

Ilustración 9. Mote Tmote Sky MTM-CM5000-MSP

[11] http://www.teco.edu/bpart/
[12] https://telosbsensors.wordpress.com/

La siguiente tabla (Tabla 2) muestra un resumen de los motes vistos en este apartado.

	Microcontrolador	MHz	V.	Sleep/ µA	Com.	km	Mesh	Temp.	Hum.	€
panStamp NRG3	CC430F5137	8-24	2-3,6	2,5	802.15.4	1 - 5	✓	✓	✓	36
WiSense WHTN115	CC1120	8-16	1,8-3,6	1	802.15.4	1	✓	✓	✓	20
BPart	BLE112	32	3	20	BLE	0,4	x	✓	✓	40
Tmote Sky MTH-CM5000-MSP	MSP430F1611	8	3	1	ZigBee	0,12	?	✓	✓	~90

Tabla 2. Comparativa de motes comerciales

2.9. Estadísticas de los incendios forestales

Si bien, se puede decir que ya estamos acostumbrados a escuchar cada año noticias sobre incendios forestales por toda España, el impacto de estos a la atmósfera es el responsable de entre el 20 - 25 % de las emisiones de CO2. Y no sólo tiene un impacto ecológico, también afecta al ámbito económico, político, social, operativo y legal [8].

La superficie total de bosques es de alrededor de 4000 millones de hectáreas en todo el mundo, lo que supone un total del 31% de la superficie terrestre del planeta. A pesar de que parezca una cantidad relativamente alta, hay que tener en cuenta que cada año se queman 13 millones de hectáreas, por lo que el número se reduce drásticamente año tras año [8].

De entre los países que conforman la Unión Europea, España es el cuarto país con mayor extensión de masa forestal, con un total de 27,5 millones de hectáreas. Podríamos estar orgullosos de nuestro ecosistema si no fuese porque, hasta 2012 ya se quemaron 7,5 millones de hectáreas de las citadas anteriormente. A lo largo del año 2017 se quemaron 67000 hectáreas de superficie arbolada, el segundo peor año del decenio. El peor año del decenio sigue siendo el año 2012, con 82854 hectáreas arboladas afectadas. De los datos registrados, el año 1994 es el más catastrófico, con una superficie arbolada quemada de 250433 hectáreas. [9]

Se puede hablar también de cifras más cercanas. En la Comunidad Valenciana, en 1993, un total del 32% de la superficie boscosa ya había sido quemada. Debido a que esta cifra ha ido aumentando, la Comunidad Valenciana se ha convertido en tierra de grandes incendios. [10]

Las causas de los incendios forestales son muy variadas, pero se pueden categorizar en dos tipos: naturales y por el hombre. Las producidas por causas naturales pueden ser debidas a rayos, volcanes, chispas del choque de piedras o espontáneas. Las producidas por causas del ser humano se pueden distribuir entre accidentales, por negligencia o intencionales. Sean cuales sean las causas, el

resultado siempre es el mismo, la devastación de numerosas hectáreas de bosques con sus consecuentes impactos medio ambientales, tanto a corto como a largo plazo.

2.10. Características de los incendios forestales

Se puede definir el incendio forestal como la propagación libre del fuego sobre la vegetación forestal. Para que esto ocurra, se deben recrear unas condiciones específicas de combustible, calor y oxígeno, además de una humedad por debajo del 25 % o la existencia de resinas inflamables. Estos tres factores son el llamado triángulo del fuego. Si alguno de estos factores no está presente, no se produce el fuego.

Los incendios forestales pueden ser clasificados como superficiales, subterráneos o de copa. Los subterráneos, como su nombre indica, se producen cuando se queman las raíces, entre otras, bajo tierra. Los superficiales se producen desde el suelo hasta una altura de 1,5 metros, y suelen derivar a incendios de copa. Los de copa se producen desde una altura de 1,5 metros hasta el punto más alto. [8]

Como datos de interés, por cada tonelada de masa forestal, en rasgos generales, se generan grandes cantidades de dióxido de carbono (CO_2) y monóxido de carbono (CO). Este es un listado aproximado [8]:

- Dióxido de carbono (CO_2) = 825 - 1450 Kg
- Monóxido de carbono (CO) = 15 - 330 Kg
- Ceniza = 2 - 90 Kg
- Hidrocarburos
- Oxidantes
- Óxido de azufre (SO_3)
- Óxidos de nitrógeno (N_xO_y)

2.11. La prevención de incendios forestales en las Smart Cities

Como ya se ha visto, las aplicaciones que tienen las Smart Cities son muy diversas. Una de ellas, la que más concierne a este proyecto, es la aplicación a la prevención de los incendios forestales.

Dado que la naturaleza de las Smart Cities es la obtención de datos, ofrecen la aplicabilidad perfecta para gestionar los datos ambientales que se pueden obtener de los bosques o parcelas. Haciendo un uso más exhaustivo y útil de estos datos, no sólo se puede extraer datos climatológicos, también se puede ayudar a vigilar nuestros bosques. Esto puede conseguirse mediante la vigilancia activa de los datos que se

extraen. Es decir, mediante un procesamiento en tiempo real de los datos extraídos, podemos concluir si se puede estar produciendo una situación de incendio o no. Con todo ello, unido a las labores de vigilancia ya existentes, se puede conseguir ampliar la visión de los vigilantes forestales y aumentar así la eficacia en la detección, que reduciría la cantidad de superficie devastada por los incendios forestales.

3. Análisis del problema

Este capítulo plantea la solución propuesta, surgida a través del análisis del problema y generando los requisitos necesarios que satisfacen todas las especificaciones extraídas. De esta forma se puede seleccionar la composición de la red que conformará el sistema.

3.1. Planteamiento del problema

Teniendo en cuenta el ámbito de aplicación de este proyecto, la instalación de la red de sensores debe ser muy poco intrusiva, ya que se busca perjudicar lo menor posible el ecosistema donde se implanta y este podría ser un sistema que afecte a la flora y fauna allá donde se instale. Es por esto por lo que la red debe constar de sensores de pequeño tamaño, situados como máximo a 400 metros unos de otros y capaces de ser instalados con facilidad e integrarse en el ambiente.

Puesto que los sensores se instalarán al aire libre, deberán ser resistentes a las inclemencias del tiempo y a los daños que pudiese provocar la fauna. A su vez, como se considera que la red de sensores estará situada en una zona desprovista de acceso a la red eléctrica, deberá ser un sistema autónomo que funcione de forma indefinida. Además, al ser zonas de difícil acceso y situadas a largas distancias, los operarios no podrán acceder a los sensores con frecuencia, por lo que deberán ser auto configurables una vez se hayan instalado en su emplazamiento final. En consecuencia, la interconexión de las balizas no puede ser cableada, ya que se consideraría una instalación intrusiva en el ecosistema, además de costosa por las amplias zonas que podría abarcar la red.

Los sensores medirán los datos de temperatura y humedad una vez cada media hora y los enviarán a través de la red. En caso de que se detecte un posible incendio, todos los sensores de la red realizarán mediciones en tiempo real, cosa que permitirá un seguimiento exhaustivo del incendio. Estos datos podrán ser de ayuda para los servicios de extinción. Una vez los sensores hayan recogido los datos, los enviarán al servidor, donde los mostrará mediante una interfaz gráfica. (Ilustración 10)

La interfaz gráfica mostrará la posición de cada sensor en un mapa y los datos recogidos por cada uno de los sensores. Cuando uno de los sensores detecte un posible incendio, se mostrará un aviso alertando de ello e informando sobre la posición de la baliza que lo ha detectado.

Debe ser un sistema económico, ya que, si abarca una zona muy extensa, el precio podría ascender demasiado. El sistema debe estar en funcionamiento las 24 horas del día, los 365 días del año y el servidor podrá estar situado en cualquier parte de la red de sensores.

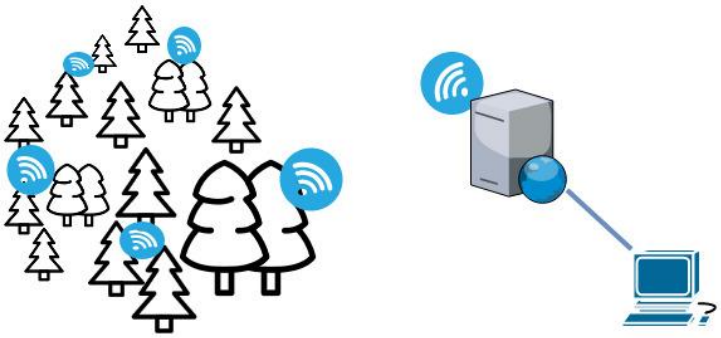

Ilustración 10. Diagrama del sistema propuesto

3.2. Obtención de requisitos

Dada la aplicación del proyecto que se plantea, el principal objetivo es la detección precoz de los incendios forestales, identificando las áreas afectadas y reforzando así, los medios de extinción existentes. Para ello, podemos clasificar los requisitos en funcionales y no funcionales. Estos requisitos se deben cumplir para poder satisfacer la completa funcionalidad del proyecto.

3.2.1. Requisitos funcionales:

El sistema deberá tener las siguientes características:

R1. Los sensores deben ser de pequeño tamaño
R2. Los sensores deben medir la humedad y temperatura
R3. Los sensores deben tomar medidas cada 30 minutos
R4. Los sensores se deben interconectar de forma inalámbrica
R5. Los sensores deben estar situados a un máximo de 400 metros unos de otros
R6. Los sensores deben tener un sistema de energía independiente, sin fuentes de alimentación externas

R7. Los sensores deben ser auto configurables, sin intervención humana
R8. Los sensores deben ser resistentes a la climatología y a la fauna
R9. Los sensores deben enviar los datos al servidor
R10. El servidor debe mostrar los datos mediante una interfaz gráfica
R11. La interfaz gráfica debe mostrar la posición de los sensores en un mapa
R12. La interfaz gráfica debe mostrar los datos de temperatura y humedad de cada sensor
R13. La interfaz gráfica debe mostrar una alerta con la posición de la baliza que haya detectado el incendio

3.2.2. Requisitos no funcionales:

R14. La red de sensores debe ser poco intrusiva
R15. Los sensores se deben instalar con facilidad
R16. El servidor podrá estar situado en cualquier punto de la red de sensores
R17. El sistema debe estar siempre funcionando
R18. El sistema debe ser económico

3.3. Especificación de requisitos

Anteriormente, se han obtenido una serie de requisitos que el sistema debía satisfacer para el problema planteado. Con ello se generan una serie de especificaciones que el sistema debe cumplir:

E1. Los motes deben ser de pequeño tamaño y versátiles
E2. Los motes deben tener sensores de temperatura y humedad
E3. Los motes deben ser económicos
E4. Los motes deben tomar datos cada 30 minutos
E5. Los motes deben tener conectividad inalámbrica para enviar los datos
E6. Los motes deben comunicarse adoptando una topología de red y un enrutamiento que minimice los costes sin que afecte al alcance de la red
E7. Los motes deben situarse a máximo 400 metros unos de otros
E8. Los motes deben alimentarse por algún tipo de acumulador de carga eléctrica
E9. Los motes deben poder recargar los acumuladores de carga eléctrica
E10. Los motes deben consumir poca energía
E11. Los motes deben administrar la red de forma automática
E12. Los motes deben ser capaces de reconfigurarse automáticamente en caso de fallo
E13. Los motes deben estar protegidos por una carcasa resistente y estanca
E14. Los motes deben integrarse con el ambiente
E15. Los motes enviarán los datos por la red hasta ser recibidos por el servidor
E16. El servidor debe estar conectado en cualquier parte de la red de sensores
E17. El servidor debe mostrar los datos por pantalla

E18. El servidor debe mostrar un mapa que sitúe los motes
E19. El servidor debe mostrar los datos obtenidos por cada mote
E20. El servidor debe mostrar una alerta con la posición de la baliza que haya detectado el incendio
E21. El sistema debe funcionar ininterrumpidamente

Los requisitos R2, R3, R4, R5, R8, R9, R10, R11, R12, R13, R16, R17 y R18 generan las especificaciones E2, E4, E5, E7, E13, E15, E17, E18, E19, E20, E21, E16 y E3 respectivamente.

Además, algunos requisitos generan más de una especificación. Es el caso del requisito R6, que se subdivide en las especificaciones E8, E9 y E10 ya que resulta necesario si se quiere tener un sistema de alimentación autónomo. El requisito R7 se subdivide en las especificaciones E11 y E12 ya que cabe diferenciar las distintas tareas que requiere una configuración automática.

Por otro lado, la unión de algunos requisitos genera especificaciones nuevas. Los requisitos R1 y R14 crean la especificación E14 porque son complementarios. Los requisitos R1 y R15 confluyen en la especificación E1 por la misma razón. Y los requisitos R4 y R18 confluyen en la especificación E5 ya que R18 influirá en R4.

3.4. Análisis de las posibles soluciones

Se puede observar que las especificaciones plantean una WSN con una estructura concreta y un funcionamiento bien definido. Para obtener las posibles soluciones, se tienen en cuenta varias especificaciones que influyen en la decisión.

3.4.1. Selección de topologías

Para la selección de la posible solución de topología se plantea mediante las especificaciones E6, E12, E15 y E16. Haciendo un simple repaso a estas especificaciones, se puede observar que existe una única solución que las cumple todas, es el caso de la topología en malla. Como se ha explicado anteriormente, es la única que permite colocar el o los sumideros en cualquier parte de la red (E15, E16) y permite una organización no-jerárquica, que ayuda a aumentar el alcance de la red (E6). Lo cual la hace ideal para que, en caso de fallo de alguno de los nodos, se opte por una ruta alternativa para el envío de datos (E12), ofreciendo una flexibilidad óptima.

3.4.1. Selección de algoritmo de enrutamiento

Con respecto a la implementación del enrutamiento de los datos, se opta por un enrutamiento por inundación, ya que este tipo de enrutamiento transfiere los datos a todos los nodos de la red. Esto permite obtener un estado del sistema desde cualquier parte de la red (E16) y a su vez, mantener las características de flexibilidad, escalamiento y confiabilidad que ofrece la topología de red en malla.

Además, profundizando más en este algoritmo y a modo de ejemplo, se puede apreciar en la Ilustración 11 que, adoptando una red en la que sus nodos se sitúan formando hexágonos regulares, se obtienen unas cualidades de propagación mejores aún. Esta formación hexagonal es idílica porque el hexágono regular es la forma poligonal con mayor número de lados que puede crear cuadrículas uniformes, es decir, sin dejar huecos sin cubrir ni crear solapamientos. De esta forma se puede aprovechar al máximo el área cubierta por los nodos y se puede conectar con un mínimo de 3 nodos vecinos.

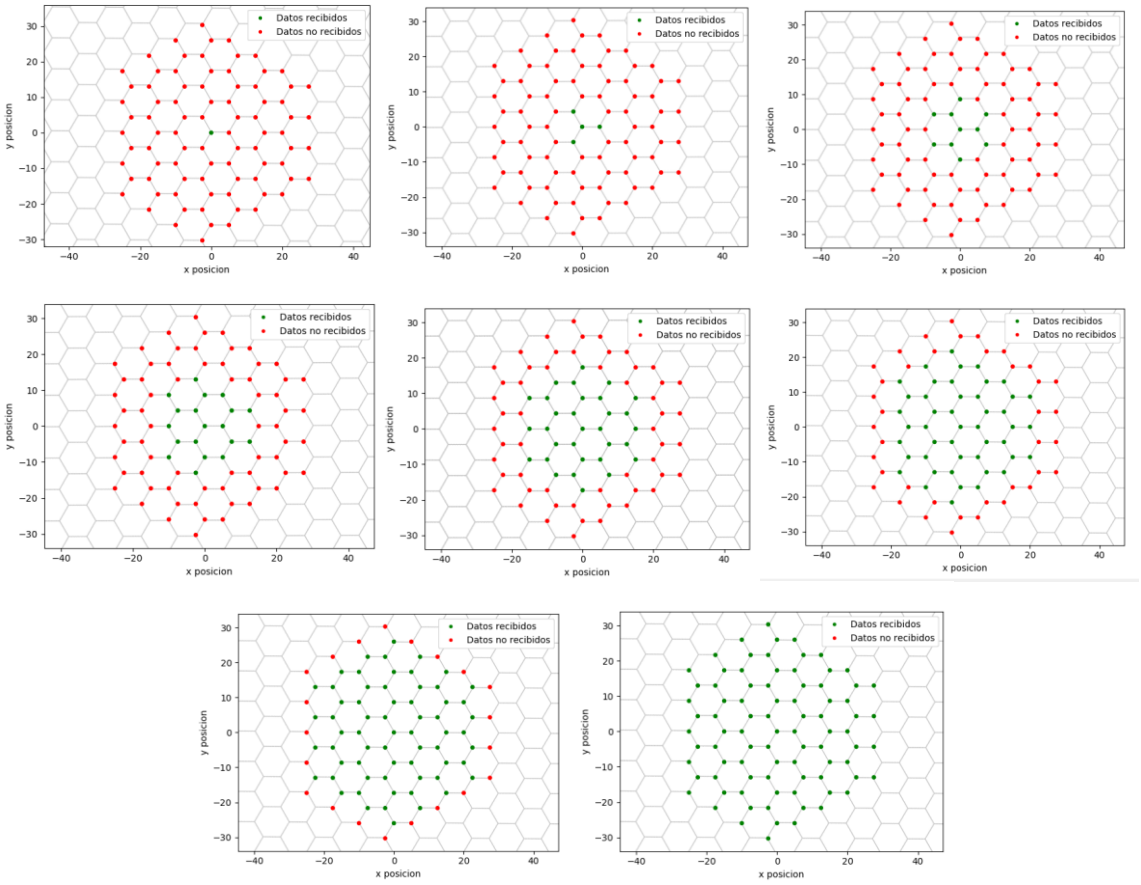

Ilustración 11. Iteraciones 1 a 8 del algoritmo de enrutamiento por inundación en redes con formas hexagonales

Con esta estructura, la propagación que sigue la relación (1) es

$$x_{n+1} = \ x_n + 3n, \tag{1}$$

donde x_{n+1} es el número de nodos de la capa siguiente, x_n el número de nodos de la capa actual y n la capa actual. Es decir, el primer nodo se sitúa en la capa 1, por lo que n será 1. De esta fórmula, se obtiene que la propagación producida de los datos será siguiendo la secuencia 1 – 4 – 10 – 19 – 31 – 46 – 64 – 85 para sus 8 primeras iteraciones del algoritmo, llegando a 109 nodos alcanzados en la iteración 9 y 1054 nodos en la iteración 27. Unas cifras bastante significativas para una red de sensores. Es muy curioso y paradójico que, mediante este algoritmo, de la sensación de que los datos se propagan da la misma forma que el fuego en un incendio.

En concepto de tolerancia a fallos, este algoritmo de enrutamiento es bastante fuerte ya que, como puede observarse en la Ilustración 12, los datos son propagados a todos los nodos a pesar de que un nodo falle, a no ser que fallen todos los nodos que actúan como puente para una sección de la red. Como se daría el caso en que se tratase de un nodo del borde exterior de la red, el cual tenga conexión con un solo nodo de esta. En dicho caso, ni siquiera existiría ruta posible.

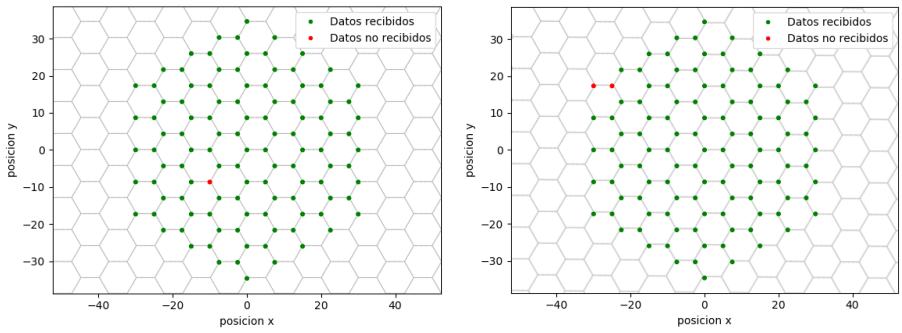

Ilustración 12. Resultado de la propagación de los datos en los casos en que falla un nodo

3.4.1. Estructura de la cadena de datos

Esta parte también es importante analizarla, ya que de la cadena de datos depende la correcta comprensión de los datos obtenidos por el sistema. Para ello, se establece una forma o estructura estándar de la cadena y así evitar datos leídos incorrectamente, falta de dígitos o incluso dígitos de más. Es por ello que se establece una posición y un rango de dígitos para cada dato, además de un dígito de identificación del dato, que ayuda a situarlo en la cadena, además de mejorar la flexibilidad de la misma para posibles modificaciones.

Encabezando la cadena, se decide poner la dirección física, también llamada MAC por las siglas de Media Acces Control, del dispositivo origen a modo de identificador, para que sea sencillo identificar a qué mote se refieren los datos. La

MAC está compuesta por 12 dígitos hexadecimales. Por lo tanto, este dato ocupa el rango de posiciones del 1 a la 12 de la cadena.

Tal como se ha explicado en el apartado de la selección del enrutamiento, es necesario un número de secuencia para diferenciar esta cadena del resto de cadenas del mismo origen. Por lo tanto, se añaden 2 dígitos numéricos que se sitúan en las posiciones 13 y 14 de la cadena.

Dado que es útil que se transmita la posición geográfica de la baliza, y así poderla situar en un mapa de forma automática (E18), se añaden también estos datos a la cadena. Para facilitar la identificación de cada dato, se usa el carácter Y para la latitud y el carácter X para la longitud. Por un lado, para la latitud, es suficiente con usar 6 dígitos, más 1 para indicar si es un valor positivo o negativo. Es decir, 7 dígitos y el identificador. Por otro lado, para la longitud es necesario un dígito más ya que, a diferencia de la latitud, la longitud llega al rango de las centenas. Es decir, 7 dígitos, más 1 dígito para indicar si el valor es positivo o negativo, más 1 dígito para el identificador. En consecuencia, la latitud ocupa el rango de posiciones del 15 al 22 y la longitud ocupa el rango de posiciones del 23 al 31 de la cadena.

Para el dato de la temperatura se usa el carácter identificador T. Dado que es interesante tener un decimal de precisión y poder medir las centenas, se obtiene una longitud de 4 dígitos más 1 para el identificador. El punto o coma para los decimales no se representa en la cadena. Las posiciones que ocupa este dato en la cadena son el rango de la 32 a la 36.

Finalmente, para la humedad se utiliza el carácter identificador H. En este caso también es interesante tener un decimal de precisión y el rango de humedad es de 0 a 100, por lo que obliga a que este dato tenga una longitud de 4 dígitos más 1 para el identificador. Al igual que en la temperatura, el punto o coma para los decimales no se representa. Este dato ocupa el rango de posiciones del 37 al 41 de la cadena.

En definitiva, la cadena de datos que se transmite consta de 41 dígitos alfanuméricos en total y está compuesta por la dirección MAC del mote, el número de secuencia, las coordenadas geográficas, la temperatura y la humedad.

3.4.2. Selección de motes comerciales

A continuación, se plantean una serie de especificaciones cruciales para los motes a modo de primer filtro, los cuales su valoración se plantea por “cumple” o “no cumple”. Las especificaciones cruciales que deben cumplir lo motes son: E1, E2 y E5, además de soportar la topología seleccionada en el apartado anterior por la especificación E6.

En este punto, la solución del mote BPart queda descartada debido a que no cumple con la especificación E6.

Una vez hecho este primer descarte, se puede observar que la mayoría de los motes sí que cumplen esos requisitos ya que son muy básicos y presentes en la mayoría de WSN. Ahora, de entre ellos, se evalúan otras especificaciones también muy deseables. A diferencia del filtro anterior, en este caso la valoración se hará por cuantificación y atendiendo a la importancia de la especificación. Las especificaciones por orden de importancia de mayor a menor son: E7, E10 y E3.

Se ha optado por tomar el consumo en modo Sleep como relevante, porque es el estado en el que estará el dispositivo la mayor parte del tiempo. Es decir, el consumo en ese modo será crucial para el tiempo de vida del mote, sobre todo cuando las condiciones climatológicas o entorno no sean favorables. El alcance también es otro de los factores que inciden en la toma de la decisión, puesto que influirá en el número de nodos utilizados por espacio y, en consecuencia, económicamente en el proyecto. Asimismo, la topología en malla utilizada también influye directamente en el alcance y el coste total del proyecto. Además de pertenecer a las especificaciones, el precio también es otro de los factores deseables, pues será crucial para que se pueda plantear su implantación y en qué magnitud.

Por el contrario, otros factores no se han tenido en cuenta ya que se desestiman su importancia para el correcto funcionamiento del sistema y no pueden resultar un inconveniente dentro de lo habitual.

Concretamente, repasando los factores anteriores en conjunto con la tabla comparativa de motes comerciales (Tabla 2) podemos decir que las posibles soluciones serían panStamp NRG3 y WiSense WHTN155, al ser las óptimas de las analizadas para el desarrollo de este proyecto.

3.4.3. Selección de componentes para un mote propio

Una mejor idea que se propone es el desarrollo de un mote específico para nuestras necesidades. Esto puede conllevar las ventajas de reducir drásticamente el coste de los motes (E3) y el tamaño (E1), al tener componentes y características ajustadas a las necesidades planteadas. A su vez, mantener las características que se requieren en la especificación de requisitos.

Para la elección del microcontrolador, en el mercado podemos encontrar varios chips que implementan Bluetooth Low Energy y Long Range (E7) con red en malla (E11, E12) y poseen modos Sleep, que les permiten ahorrar energía cuando no realizan ninguna función(E10). Además de la peculiaridad de despertar del modo Sleep mediante una señal de reloj, más concretamente, RTC, que complementa el ahorro energético. A continuación, se analizan algunos de ellos:

EFR32BG13[13]: Fabricado por Silicon Labs. Se trata de un chip BLE y BLR que soporta topología en malla. El microcontrolador se trata de un ARM Cortex-M4 con una velocidad de 40 MHz, RAM de 64 kB y una memoria Flash de 512 kB. Funciona a 3V, en modo Sleep consume 1,3 µA. Su programación es con lenguaje C mediante Silicon Labs Bluetooth SDK. Su precio oscila alrededor de los 5 €.

nRF52840[14]*:* Fabricado por Nordic Semiconductors. Es otro chip BLE y BLR que también soporta topología en malla. Implementa microcontrolador + radiotransmisor. El microcontrolador se trata de un ARM Cortex-M4 con una velocidad de 64 MHz. Posee una RAM de 256 kB y una memoria Flash de 1MB. Funciona a 3V y en modo Sleep consume 3,16 µA. Se programa en lenguaje C mediante el uso de Nordic SDK. Su precio oscila los 7 €.

BL654[15]: Fabricado por Laird Connectivity. Está compuesto por un nRF52840, pero con características optimizadas, por lo que tiene casi las mismas características. Microcontrolador ARM Cortex-M4 de 64MHz, con 256kB de RAM y 1MB de memoria Flash. Amplía la gama de interfaces de comunicación disponibles. Otra de sus características es que incorpora una librería que le permite trabajar con sencillos comandos AT o programar mediante su propio lenguaje de programación smartBasic, que se trata de una implementación del lenguaje BASIC, optimizado para sistemas de memoria reducida. Dado que está formado por un chip nRF52840. también permite la programación en lenguaje de programación C mediante Nordic SDK. Funciona a 3V y en modo Sleep consume 3,1 µA. Su precio oscila los 9 €.

Como resultado de los microcontroladores analizados, se escoge utilizar el BL654 del fabricante Laird Connectivity ya que, a pesar de tener un precio ligeramente superior, posee unas características muy valiosas para este proyecto, como son la posibilidad de trabajar con comandos AT o la programación mediante smartBASIC que, junto con Nordic SDK, le confieren una versatilidad superior al resto de microcontroladores analizados. Característica que hace disminuir el tiempo de aprendizaje para la programación del microcontrolador.

Otros componentes que conforman el mote son un sensor de temperatura y un sensor de humedad. Adicionalmente, se pueden añadir otros sensores como un sensor de CO2 o CO, entre otros. Arbitrariamente, se escoge el uso del sensor *SHTW2* de *Sensirion AG* que, con una tensión de 2 voltios y un consumo muy reducido, integra el sensor de temperatura y humedad en el mismo chip (E2). Además,

---

[13] https://www.silabs.com/products/wireless/bluetooth/blue-gecko-bluetooth-low-energy-socs/device.efr32bg13p632f512gm48
[14] https://www.nordicsemi.com/Products/Low-power-short-range-wireless/nRF52840

[15] https://www.lairdconnect.com/wireless-modules/bluetooth-modules/bluetooth-5-modules/bl654-series

a razón del medio en que se encuentran los motes, la solución ideal que se plantea para el suministro de energía es una placa solar de 55mm x 60mm, capaz de generar 3v 150mA (E9), que se complementa haciendo uso de dos pequeñas pilas de botón recargables CR927 de 3V 30mAh, carga suficiente para hacer funcionar el microcontrolador y almacenar energía para cuando la luz solar sea insuficiente (E8).

Componentes, como la batería, la placa solar o los sensores, no son analizados en este proyecto por la falta de conocimientos y la gran variedad que existe de ellos. Es por ello que su elección es arbitraria y no se han tomado referencias.

3.4.4. Características de la carcasa

Con respecto al diseño de la carcasa protectora de los motes, no concierne su realización al ámbito de este proyecto. Pero igualmente se plantea, tal como se muestra en la Ilustración 13, un diseño preliminar de la misma en lo que a forma y color se refiere.

Dicha carcasa posee una forma que permite maximizar o mejorar el ángulo con el que la luz solar incide sobre la placa solar y así poseer un mejor suministro de energía. A su vez, la ausencia de superficies horizontales impide que ciertos animales se posen sobre ella, obstruyendo el paso de luz o deteriorándola. En la parte inferior, posee una trampilla desde a la que se accede al interior y que tiene unas finas aberturas que permite la circulación del aire al interior para mejorar las lecturas tomadas por los sensores, a la vez de protegerlos de las inclemencias del tiempo(E13).

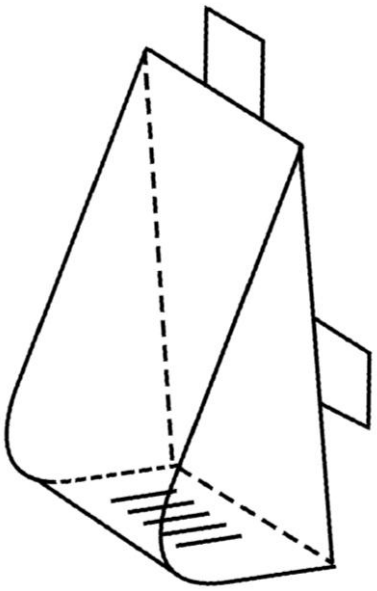

Ilustración 13. Boceto de la estructura y forma de la carcasa

Con respecto al color, puede ser una tonalidad marrón, parecida a la de los troncos donde se vaya a situar o incluso con algún patrón imitación a madera(E14).

El material del que se compone es de plástico (E13), lo cual hace que sea factible para la producción con moldes o incluso con impresoras 3D.

3.4.5. Configuración del servidor

Dado que se trata de una red en malla y que los datos son enrutados por inundación, no existe un nodo destino concreto. Por ello, el servidor actúa como otro

nodo integrante de la red que, a diferencia de los motes, éste no envía datos, sólo espera a la escucha de los datos que circulan por la red.

Por lo tanto, el servidor, obteniendo los datos desde un nodo auxiliar, los trata para mostrarlos en pantalla. Los datos son transformados en información útil, se extrae las coordenadas, temperatura y humedad, y son posicionados en un mapa (E17, E18, E19, E20) que evalúa y alerta sobre la posibilidad de incendio si la humedad es inferior a 30% y la temperatura es superior a 60 ºC.

3.5. Solución propuesta

A modo de resumen, se plantea la siguiente propuesta elegida en los puntos anteriores para el desarrollo del sistema planteado:

La arquitectura del sistema se construye mediante una red con *topología en malla* y con *enrutamiento por inundación*. Lo cual maximiza el alcance y la flexibilidad de la red, parte muy importante en este sistema y que, además, permite conocer el estado del sistema desde cualquier punto del que se conecte.

El sistema se compone de *motes híbridos*, que son capaces de tomar los datos de los sensores, pero también enrutar los datos provenientes de otros motes. Dichos datos son enviados de forma inalámbrica por la red, mediante conectividad *Bluetooth* en su versión *Low Energy Long Range*, para así dotar a la red de una facilidad de conexión con cualquier dispositivo móvil, ampliamente integrado en todo el mundo.

A su vez, los motes se componen de un microcontrolador + radiotransmisor *BL654* de *Laird Connectivity*. Ideal para este propósito ya que integra el microcontrolador y el radiotransmisor bluetooth en el mismo chip, ahorrando energía y espacio. Además de contar con la facilidad de programación mediante smartBASIC y comandos AT.

Los sensores integrados en los motes son, principalmente de temperatura y humedad. Mediante el uso del componente *SHTW2* de *Sensirion AG*, se integra el sensor de temperatura y el sensor de humedad en un mismo componente.

El mote se alimenta por *dos pilas de botón recargables CR927* de 3V 30mAh, complementadas por una placa solar de 55mm x 60mm que genera 3v y 150mA.

Para la carcasa, se plantea un diseño personalizado, fácil de producir y con una alta integración en el ambiente (Ilustración 13).

El servidor se compone de un nodo que está a la escucha de los datos que circulan por la red y los transmite a un ordenador, el cual los muestra por pantalla mediante un programa de interfaz sencilla.

Por último, se establece una forma estándar para la cadena de datos enviados por los motes. Esta cadena consta de 12 dígitos para la MAC de origen, 2 dígitos para el número de secuencia, 7 dígitos para la coordenada de latitud, 8 dígitos para la coordenada de longitud, 4 dígitos para la temperatura, 4 dígitos para la humedad y 4 dígitos para la calidad del aire. Además de un dígito extra para cada dato, a excepción de la MAC y el número de secuencia, que permite identificar cada dato en la cadena y de esta forma incrementar la flexibilidad en caso de querer añadir más datos en la misma.

3.6. Solución alternativa

Por otra parte, también se puede optar por adquirir los motes ya desarrollados, aunque esto implica que se puedan perder algunas características del sistema exigidas por las especificaciones. En este caso, y dadas las cualidades de los motes comerciales vistos anteriormente y sus características (Ilustración 11), la mejor opción que se opta, de entre los analizados, es el WiSense WHTN155. Dicho mote soporta una topología de red en malla, posee sensor de temperatura y humedad, y su bajo consumo y bajo precio lo hacen una opción muy competitiva.

3.7. Coste del sistema

El coste del sistema también es una parte importante no sólo de este proyecto, sino de cualquier proyecto que se plantee. Un muy buen proyecto puede quedar totalmente descartado sólo por el hecho de no ser rentable o, simplemente, necesitar un presupuesto demasiado elevado. El hecho de que un proyecto sea asequible o no siempre es una apreciación subjetiva y es por ello que en este apartado se trata de esclarecer cuanto podrían ser los costes de implantar este proyecto para desplegar la red de sensores planteada.

Sabiendo los componentes seleccionados de los que se componen los motes, es tan sencillo como hacer una suma de los costes de cada uno y así obtener un coste aproximado. De esta forma, sabiendo que el precio aproximado del BL654 es de 9,00 €, del sensor SHTW2 es de 2,70 €, de las pilas recargables CR927 es de 2x 0,08 €, de la placa solar 2,50 € y de una pequeña PCB (Printed Circuit Board), placa para soldar los componentes, es de 0,25 €. Se descarta el precio que puede tener fabricar la carcasa, que podría ser de unos 3 a 5 €. En total cada mote tiene un coste aproximado de 14,61 €, 18 € con la carcasa personalizada (Tabla 3). Cabe tener en cuenta que dicho coste se ve reducido por la fabricación en grandes cantidades, por lo que podría ser incluso menor.

COMPONENTE	COSTE
1 X BL654	9,00 €
1 X SHTW2	2.70 €
2 X CR927	0,16 €
1 X PLACA SOLAR	2,50 €
1 X PCB	0,25 €
TOTAL	14,61 €

Tabla 3. Cálculo del coste de cada mote

Suponiendo que los motes se sitúan en una formación hexagonal, a la que se procede a llamar *celda*, compuesta de 6 motes situados en sus vértices. Cada lado del hexágono mide 250 metros. En su conjunto, el rango de alcance que forman es prácticamente el de un círculo, un círculo de 500 metros de radio. Por lo tanto, el área de este círculo será de 785000 metros cuadrados, que equivale a 0,785 kilómetros cuadrados. Por lo tanto, se obtiene como resultado que la primera celda tiene un coste de 81,66 € y abarca un área de 0,785 kilómetros cuadrados.

Para las siguientes celdas que se incorporen a esta primera, se reduce el coste y el alcance, ya que las áreas se solapan y los vértices coinciden. A grandes rasgos se puede decir que, si la celda que se añade sólo tiene una celda vecina, se añadirían sólo 4 motes. Si tiene dos celdas vecinas, se añaden 3 motes. Y sucesivamente de forma inversamente proporcional al número de celdas vecinas.

Esto sería el coste de la implementación del sistema, pero para una primera implementación hay que tener en cuenta los costes del desarrollo también. Es decir, para generar el código en este sistema, habría que hacerlo mediante el uso de unos kits de desarrollo, llamados Development Kit (DK). Más concretamente se necesitan tres de estos dispositivos para poder desarrollar y probar una red de prueba. Esto conlleva unos gastos más elevados, ya que estos dispositivos son más completos, pero mucho más caros que únicamente el chip suelto. Para el microcontrolador elegido ha sido creado el BL654 DVK, que tiene un precio aproximado de 65 € sin impuestos ni gastos de envío. Esto establece un coste de desarrollo superior a 195 €, cosa que incrementa en gran medida el presupuesto para su desarrollo y prueba.

4. Diseño

Los diagramas permiten una representación de la información más fácil de entender. En este capítulo se muestran y describen los diseños que conforman la solución del sistema planteado.

Para la realización de los diagramas, se ha hecho uso de las herramientas gratuitas Draw.io[16], Fritzing[17] y Moqups[18].

4.1. Diagrama del sistema

El diagrama presentado en la Ilustración 14. Diagrama del sistema, representa las relaciones entre las entidades del sistema. Se puede ver que una WSN está compuesta de motes y que a su vez un servidor puede conectarse a ella, en este caso, para obtener los datos. Los motes están compuestos de sensores, una unidad computacional, un radiotransmisor, un acumulador de carga y una placa solar. La placa solar recarga el acumulador de energía. El acumulador de energía alimenta la unidad computacional y el radiotransmisor. El sensor envía datos a la unidad computacional, a su vez esta intercambia datos con el radiotransmisor.

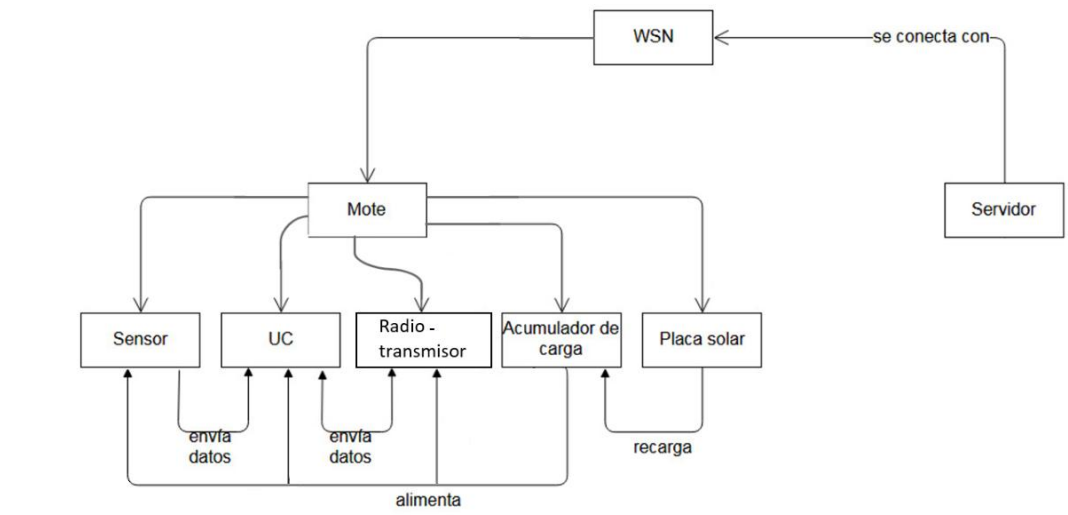

Ilustración 14. Diagrama del sistema

[16] https://www.draw.io/
[17] http://fritzing.org/home/
[18] https://moqups.com/

4.2. Diagrama de Casos de Uso

El diagrama de casos de uso representa las acciones que pueden realizar cada actor del sistema (Ilustración 15). El sistema está compuesto por dos actores, el Usuario y los Motes. El Usuario únicamente puede ver los datos de los sensores. Por otro lado, el Motes puede realizar las acciones de dormir, despertar, recibir datos, enviar datos y medir temperatura. Esta última acción realiza consecutivamente la acción de medir humedad.

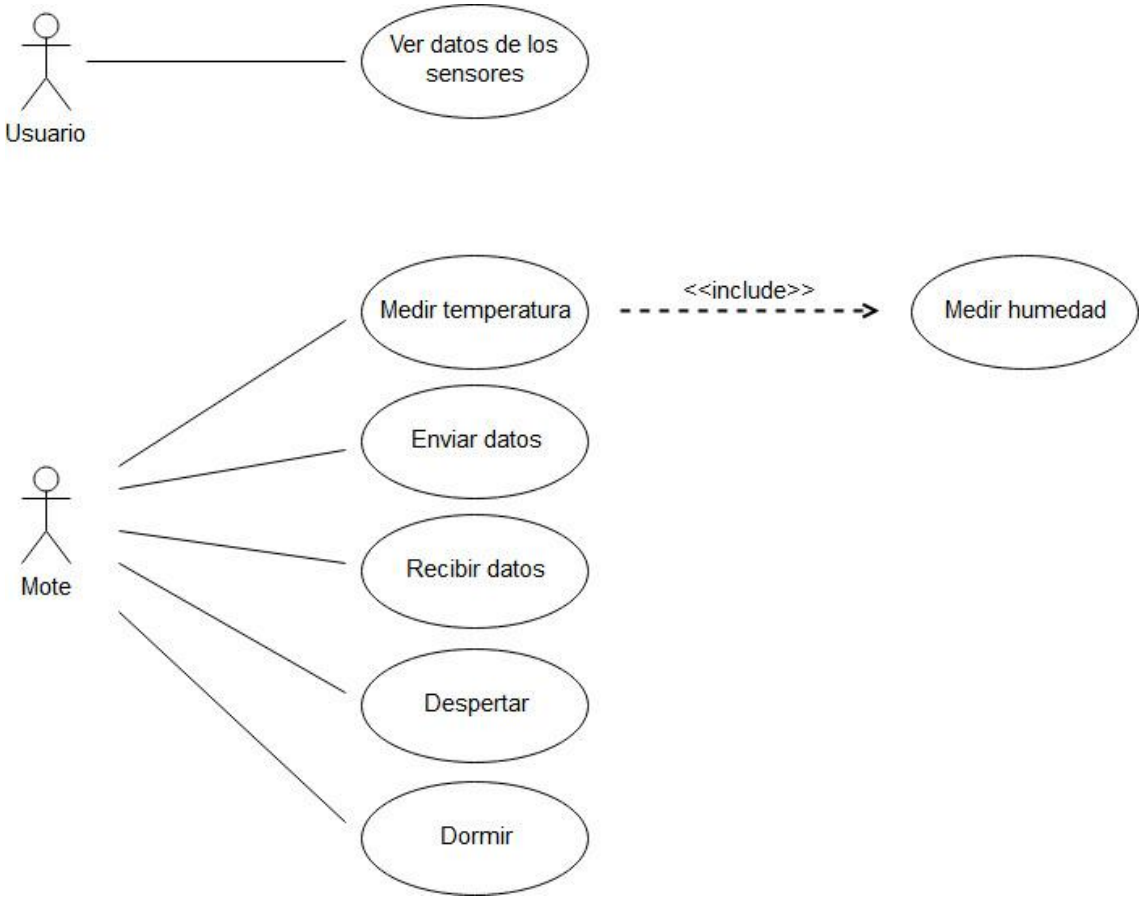

Ilustración 15. Diagrama de casos de uso del sistema

4.3. Diagrama de flujo del algoritmo de los motes

El diagrama de flujo de la Ilustración 16 representa el algoritmo que ejecutan los motes y que les permite recibir datos, reenviarlos y enviar los suyos propios. Puede observarse también que, mediante la generación de un número aleatorio y un bucle, se consigue que los motes envíen sus datos en momentos distintos unos de otros y en periodos de tiempo distintos.

A modo de breve explicación, cuando el algoritmo se inicia, comienza inicializando el bluetooth y se conecta a la red. Seguidamente genera un número aleatorio y ejecuta un bucle. En este bucle se comprueba si el número de iteración coincide con el número aleatorio. En caso de que coincida, se genera la cadena y se ejecuta la rutina o método difundir. Al terminar el método se pasa a la siguiente iteración del bucle. En caso de que el número aleatorio no coincida, se consulta si hay algún dato entrante al bluetooth. Si hay un dato entrante, se recibe la cadena, se comprueba si es una réplica. Si se trata de una réplica, se descarta y se continúa la ejecución. En caso contrario se ejecuta el método difundir y se pasa a la siguiente iteración del bucle. Si no hay ningún dato entrante, se continúa la ejecución del algoritmo. Entonces, se comprueba si la iteración del bucle es la última. En caso de que lo sea, se genera un nuevo número aleatorio y se inicia de nuevo el bucle. En caso de que la iteración no sea la última, se leen los datos de los sensores y se comprueba si pudiera estar detectando una situación de alerta. Finalmente se pasa a la siguiente iteración del bucle.

Para la rutina Difundir tenemos el parámetro Cadena. Comienza escaneando los dispositivos cercanos y, para cada dispositivo encontrado, se comprueba si es el origen de la cadena o el dispositivo anterior que se la envió. En caso de que sea el dispositivo origen o anterior, pasa al siguiente dispositivo de la lista. En caso contrario, envía la cadena al dispositivo y pasa al siguiente. Una vez recorrida toda la lista de dispositivos, termina la ejecución de esta rutina.

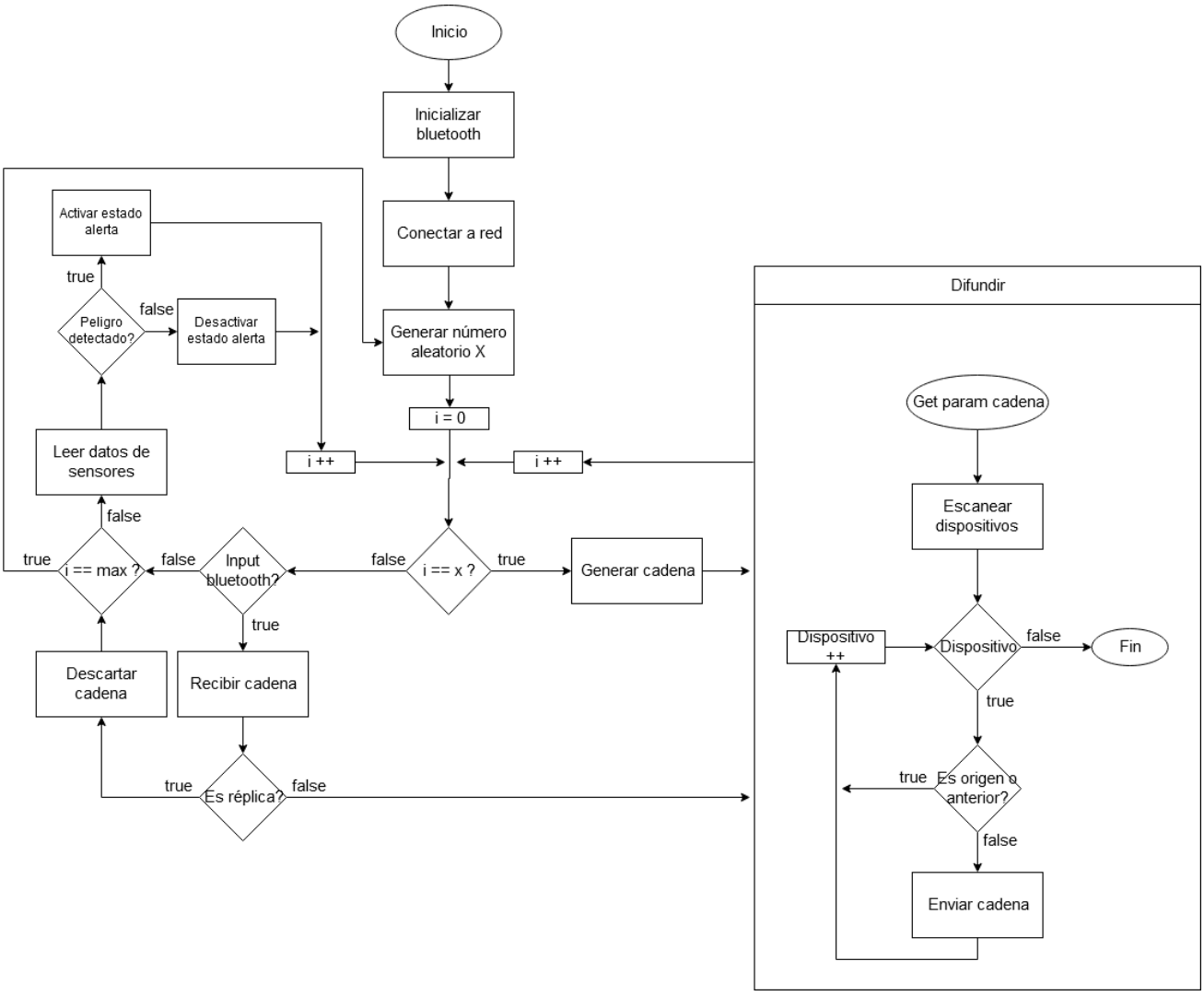

Ilustración 16. Diagrama de flujo del algoritmo de los motes

4.4. Diagrama de flujo del algoritmo del servidor

El diagrama de flujo de este apartado (Ilustración 17) representa el algoritmo que se utiliza en el servidor. En él, también podemos observar que se trata de un bucle infinito esperando las recepciones de cadenas.

En primera instancia se inicializa el bluetooth, se conecta a la red de sensores y espera a que los motes le envíen cadenas de datos. Cuando recibe una, comprueba si se trata de una réplica. Si es una réplica, descarta la cadena y vuelve a la espera de otra recepción. En caso negativo comprueba si el mote ya está registrado en el programa. Si el mote ya estaba registrado, actualiza sus datos y continúa el flujo del algoritmo. Si, por el contrario, es la primera vez que se tienen datos de este mote, se guardan sus datos, se muestra en el mapa y se continúa el flujo. A continuación, se evalúa si los datos recibidos de temperatura muestran una situación de peligro. Si lo es, se muestra el peligro por pantalla. En caso negativo, se evalúa si pudieran indicar una situación de alerta. Si puede ser una situación de alerta, se muestra la alerta por pantalla, si no se vuelve a la espera de una nueva cadena de datos.

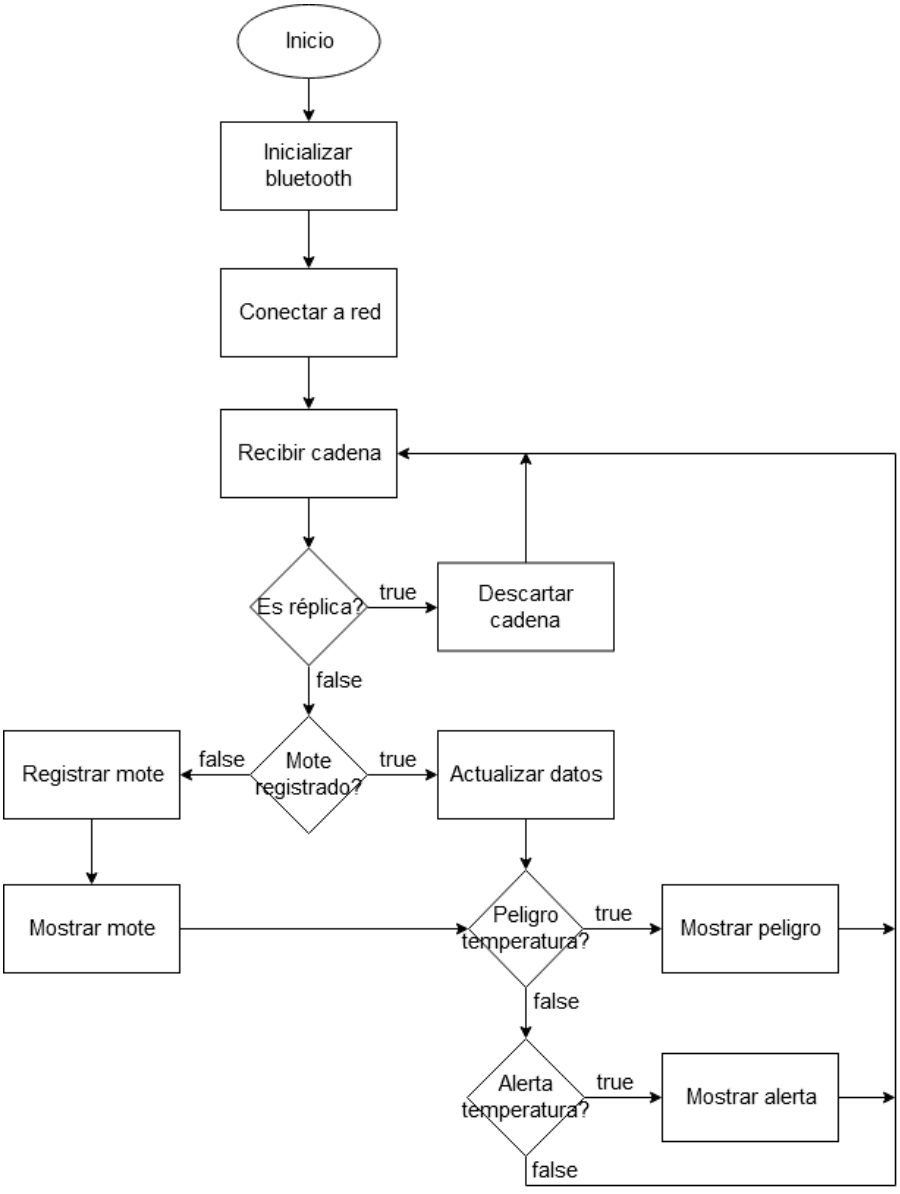

Ilustración 17. Diagrama de flujo del algoritmo del servidor

4.5. Diseño de la interfaz gráfica

Se diseña este boceto (Ilustración 18) como orientación del aspecto de la interfaz gráfica del servidor. Se presenta una interfaz gráfica simple y sin ventanas emergentes, basada en un diseño de tres bloques. Un módulo central en el que se muestra el mapa donde van situadas las balizas. Un módulo derecho donde aparecen datos de utilidad como los datos de la baliza seleccionada o los mensajes que llegan por el puerto serie. Y un módulo inferior donde se muestra el estado del sistema mediante un icono y una caja de texto. En la parte superior, se sitúa una botonera donde elegir el puerto de serie de escucha.

Ilustración 18. Boceto de la interfaz gráfica del servidor

4.6. Diseño de la carcasa

En apartados anteriores ya se ha descrito las características de la carcasa. Como se puede observar, se ha elegido un motivo de corteza de árbol y se ha situado la antena en la parte superior. A continuación, se muestra un modelo más elaborado de la misma y su aspecto integrada en un ambiente de ejemplo (Ilustración 19 e Ilustración 20).

Ilustración 20. Diseño de la carcasa de los motes

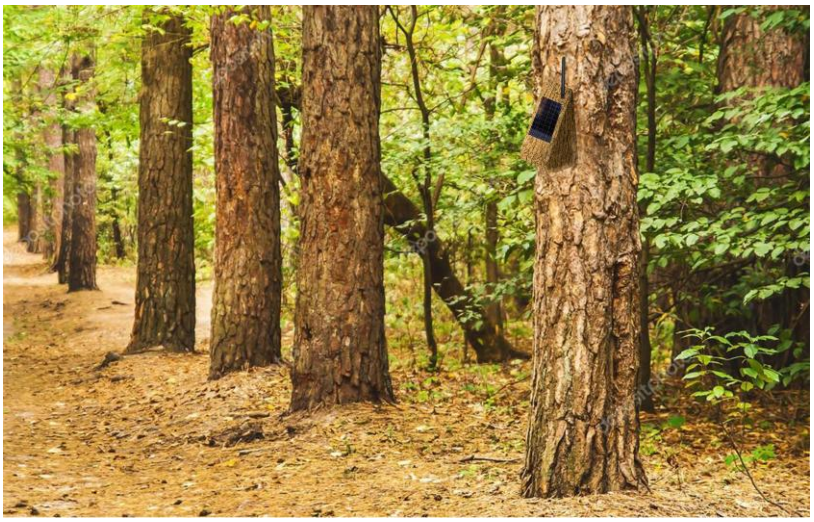

Ilustración 19. Muestra de la integración de la carcasa en el ambiente

5. Implementación

En este capítulo se describe la implementación de la solución propuesta. Mediante la selección de los componentes y de las herramientas de trabajo, se pretende detallar el trabajo realizado y así poder reproducirlo. Al final del capítulo, se comentan las eventualidades surgidas en el desarrollo de la implementación para poder tenerlas en cuenta en futuros proyectos o en la mejora de este.

5.1. Selección de componentes

Para el desarrollo del proyecto se comienza con la programación de los motes, que serán la principal fuente de datos para probar el resto del sistema y más concretamente el algoritmo desarrollado. Dado que no se cuenta con el tiempo ni los recursos necesarios para la implementación del sistema con los componentes propuestos, se propone como alternativa realizar un prototipo de bajo coste e interfaz amigable. Esto permite construir el prototipo en poco tiempo y comprobar la viabilidad, no sólo del sistema, sino del proyecto.

Dadas las especificaciones y las características del sistema, se opta por utilizar la plataforma Arduino[19], ya que se trata de una plataforma libre con un amplio abanico de accesorios sensores y actuadores compatibles y de fácil programación.

Para sustituir los componentes propuestos en la solución, se utiliza una placa Arduino UNO y un módulo bluetooth HM-10 en lugar del BL654. En sustitución del sensor de temperatura y humedad SHTW2, se hace uso del módulo DHT22. Con respecto a la placa solar y la batería, no se aplica en este prototipo, en su lugar, se usan baterías de 5V. Adicionalmente, se provee a los motes de un sensor de calidad del aire MQ-135 que aporta más información a los datos obtenidos por la red y razón por la cual habrá que incrementar la cadena de datos en 5 caracteres.

La placa Arduino UNO actúa como núcleo del mote, ya que es la unidad que obtiene y almacena los datos de los sensores y envía las órdenes al módulo bluetooth para su envío por la red. Posee un microcontrolador ATmega328P de 20 MHz, con una memoria RAM de 2Kbytes y una memoria Flash de 32Kbytes. Funciona a 5V y posee diversas opciones de entrada y salida de datos, entre ellas analógicas, digitales, serie, SPI o I2C, mediante sus conectores, también llamados pines.

El módulo bluetooth HM-10[20], se trata de un radiotransmisor de Bluetooth 4.0 Low Energy pero sin soporte para topología de red en malla. Como consecuencia, se realiza una implementación que simula una red en malla mediante conexiones y desconexiones continuas. Funciona a 3V e implementa una conexión serie.

El módulo sensor de temperatura y humedad DHT22 se alimenta a 3V y posee una salida de datos digital. Posee la librería DHT[21] para su uso en Arduino.

El módulo sensor de calidad de aire MQ-135, es un sensor analógico sensible al CO2, CO, NH4, Alcohol, Tolueno y Acetona. El inconveniente que posee es que las medidas de esos gases las realiza en conjunto, por lo que los datos mostrados son un conjunto de ellos, llamado calidad del aire. Los rangos de valores que muestra se clasifican por: normales si está entre 0 y 150, dañino si está entre 151 y 300 y peligroso si es superior a 300. La tensión de alimentación es de 5V y posee una salida de datos analógica y otra digital. Para su uso en Arduino se utiliza la librería MQUnifiedsensor[22].

[19] https://www.arduino.cc/
[20] https://wiki.makespacemadrid.org/index.php?title=M%C3%B3dulo_HM-10
[21] https://www.arduinolibraries.info/libraries/dht-sensor-library
[22] https://www.arduinolibraries.info/libraries/mq-unifiedsensor

5.2. Selección de herramientas

Los dispositivos de desarrollo comentados anteriormente son utilizados junto con un Integrated Development Environment o IDE, generalmente específico, que facilita la programación y permite al desarrollador hacer uso de todo el potencial del dispositivo.

Arduino, que se trata de una plataforma integrada por una gran comunidad, posee su propio IDE, Arduino IDE[23] versión 1.8.9. Este IDE posee una interfaz minimalista y sencilla, que permite no sólo compilar el código, sino integrarlo en los microcontroladores. Además, permite la integración en el IDE de nuevas librerías, placas y drivers mediante un buscador. Simple pero efectivo. Como desventaja, no ofrece autogeneración ni sugerencia de código. El lenguaje de programación que utiliza es Arduino, que básicamente es una implementación acotada de Java, por lo que su utilización es bastante familiar.

También se hace uso del IDE Processing[24] versión 3.5.3, en este caso, para mostrar por pantalla de una forma gráfica y amigable, los datos recogidos por el servidor mediante Arduino. Su interfaz es prácticamente igual que la de Arduino IDE pero con ligeras diferencias. Processing también es el nombre del lenguaje de programación que usa este IDE y, al igual que Arduino, está basado en Java.

Junto con Processing se utiliza la herramienta G4P GUI Builder[25] creada por Peter Lager. Se trata de una herramienta que se integra con el IDE de Processing y que permite construir interfaces sencillas mediante la técnica *drag and drop*, también conocida como arrastrar y soltar. Aunque genera el código de forma automática, es recomendable repasar la documentación de referencia para poder utilizar ciertas características que no se ofrecen mediante su interfaz.

5.3. Desarrollo del código de los motes

Siguiendo el diagrama del algoritmo representado en la sección anterior, se desarrolla el código (Anexo 1. Código Arduino del algoritmo) del programa en Arduino siguiendo el siguiente flujo de instrucciones:

[23] https://www.arduino.cc/en/Main/Software
[24] https://processing.org/download/
[25] http://lagers.org.uk/g4ptool/index.html

En primera instancia, se configura el módulo bluetooth, definiendo parámetros como el nombre del dispositivo, la contraseña de conexión, el rol adoptado, la visibilidad, etc. Seguidamente se almacena la dirección MAC del módulo bluetooth en una variable que, más tarde se usará como ID para generar su propia cadena. También se obtiene un valor aleatorio de un pin analógico, para utilizarlo como semilla para generar un conjunto de números aleatorios.

Seguidamente, se inicia un bucle donde el uso del número aleatorio es crucial para el correcto funcionamiento del algoritmo. El bucle se compone de una estructura condicional IF donde si la iteración del bucle coincide con el número aleatorio se inicia la rutina de envío de los datos tomados por los sensores mediante la ejecución del método “Difundir” y “Generar cadena”. En caso contrario se consulta al módulo bluetooth si tiene algún dato en el buffer a la espera de ser leído. En caso positivo, se inicia la rutina de recepción de datos mediante la ejecución del método Recibir. En caso negativo se realiza la toma de datos por los sensores y se almacenan en las variables para poder usarlos en las siguientes iteraciones del bucle. Además, si de los datos tomados por los sensores se percibe una situación de peligro, como una temperatura o índice de calidad de aire por encima del nivel de alerta, entra en estado de alerta.

El estado de alerta consiste en un reajuste del número de iteraciones del bucle para que se incremente el envío de datos, pero no demasiado para no afectar al reenvío de datos del resto de dispositivos cercanos. Cuando se activa el estado de alerta, se realiza un envío de datos anticipado. Cuando se desactiva el estado de alerta, se reestablece el bucle a su estado inicial.

En el método “Generar Cadena” se genera la cadena a transmitir, dando un formato y posicionando los datos en el orden correcto.

En el método “Recibir”, se comprueba que el dato recibido es la cadena de datos y se realiza la extracción de la MAC de origen y el número de secuencia. Seguidamente se comprueba si el número de secuencia coincide con los números de secuencia almacenados de cadenas que llegaron anteriormente, previniendo de generar ciclos en el enrutamiento. Si ya existía, la cadena se considera una réplica y se descarta. Pero si no, se almacena el número de secuencia y se inicia la rutina o método “Difundir” con la misma cadena que le ha sido enviada.

El método “Difundir” es algo más complejo y largo. Primero envía los comandos necesarios al módulo bluetooth para que comience la búsqueda de dispositivos a los que puede conectarse. El módulo bluetooth responde con el listado de dispositivos encontrados. De este listado, se extrae la dirección MAC de los dispositivos con nombre “firesensor”, que serán los dispositivos de nuestra red y los cuales son elegidos para enviar los datos. De ellos se descarta el nodo anterior, es decir, desde el que se ha recibido la cadena, y el nodo origen. Seguidamente, se envía al módulo bluetooth el comando de conexión con cada una de estas MACs escogidas y a continuación envía la cadena a transmitir y cierra la conexión.

5.4. Desarrollo del código del servidor

El código desarrollado para el servidor se puede dividir en dos partes. Una es el código desarrollado en Arduino para el hardware y la otra es el código desarrollado en Processing para el Software.

5.4.1. Código del hardware

Tal como se ha especificado en apartados anteriores, el servidor es un nodo sin las capacidades de tomar datos de los sensores ni enviar datos, únicamente espera, a la escucha de los datos que circulan por la red y que va recibiendo. Por esto, el código desarrollado para la escucha de datos en el servidor es, en realidad, una parte del desarrollado para los motes. Los detalles del código se explican en el apartado anterior, por lo que no se va a profundizar en la explicación del mismo, sólo las diferencias.

Para este nodo, se mantiene la parte de configuración del módulo bluetooth y, seguidamente, comienza un bucle infinito que espera conexiones entrantes al bluetooth para la recepción de los datos. Una vez existe una conexión, se llama a la rutina o método Recibir. Éste recibe la cadena y, al igual que hacen los motes, comprueba el nodo origen y nodo anterior para descartarlo en caso necesario. En caso contrario, y a diferencia de los motes, la envía por el puerto de comunicación serie para ser procesado por el programa desarrollado en Processing.

5.4.2. Código del software

Primero, se prepara e inicializa el entorno gráfico (Ilustración 21) con la interfaz tal como se ha establecido en apartados anteriores. El programa queda a la espera de que sea pulsado el botón que activa el nodo, iniciando el código programado en el dispositivo. En este momento se comienza un bucle en el que, al recibir los mensajes del dispositivo por el puerto serie, se van mostrando en la sección Input Log. Además, estos mensajes son analizados para identificar los que son cadenas con los datos de los motes. Cuando detecta una de estas cadenas, la subdivide y extrae cada uno de los datos que contiene, como son la dirección MAC, el número de secuencia, la latitud, la longitud, la temperatura y la humedad. Una vez con estos datos, los almacena, muestra el nodo situándolo en el mapa y analiza para comprobar si puede existir una condición de alerta o de peligro. Si la temperatura es superior a 60 grados, o el índice de calidad del aire está por encima de 400, el nodo se considera en peligro y el programa pasa a estado de alerta. Si no, si el índice de calidad del aire está entre 250 y 400, el nodo se considera en alerta, pero se mantiene el programa en estado normal. Si no se dan ninguna de estas condiciones, el nodo se considera en estado normal. Seguidamente se comprueba el estado de los motes, es decir, si por cualquier razón han pasado más de 15 minutos sin recibir datos de algún mote, este es considerado como desconectado y se da la posibilidad de ser eliminado de pantalla.

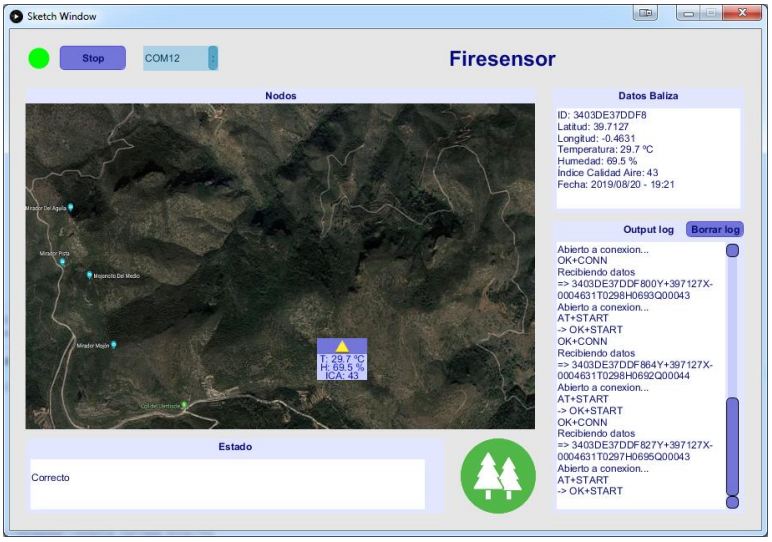

Ilustración 21. Diseño final de la interfaz

Además, el programa ofrece otras funcionalidades, como mostrar los datos recibidos del mote en la sección de Datos al presionar sobre el icono del mismo. Ver los mensajes de alerta del sistema en la sección Estado y el icono correspondiente. Incluso también permite la selección del puerto serie desde el que se tiene conexión con el hardware.

5.5. Desarrollo de las interconexiones del mote

Arduino UNO contiene una cantidad considerable de conectores, con lo cual no hay ningún problema a la hora de elegir a qué conector conectar cada uno de los módulos sensores o bluetooth. En este caso, tal como se muestra en la Tabla 4. Conexiones de los componentes del mote, la configuración de conexiones (Ilustración 22 e Ilustración 23) del Arduino que se sigue es la siguiente:

PIN ARDUINO	PIN HM-10	PIN DHT22	PIN MQ-135
2	BRK	-	-
4	-	Out	-
11	TXD	-	-
12	RXD	-	-
A0	-	-	A0
GND	GND	GND	GND
5V	-	VCC	VCC
3.3V	VCC	-	-

Tabla 4. Conexiones de los componentes del mote

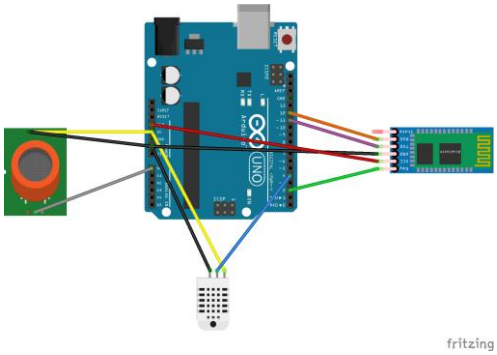

Ilustración 22. Esquema de conexiones del prototipo

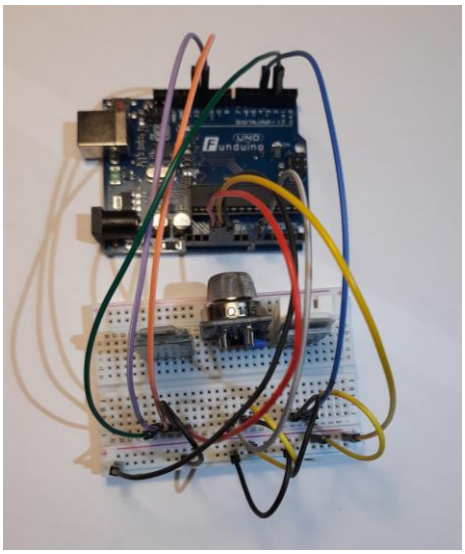

Ilustración 23. Foto de las conexiones del prototipo

5.6. Eventualidades encontradas

Para la implementación del dispositivo, se comenzó con la compra de una placa de desarrollo Bluegiga DKBLE[26] para el desarrollo del mote con el chip BLE121LR. La compra tuvo un coste de 200€. Mas tarde fue evidente que el dispositivo no cumplía los requisitos que se necesitaban, por lo que hubo que devolver el dispositivo.

También se probó el dispositivo CSR µEnergy Development Kit. Se trata de un kit de desarrollo especial para BLE con CSRmesh. Este kit está compuesto por 4 nodos programables, con un chip CSR101X que soporta la topología en malla, pero no tiene BLR. Aunque resulta un kit muy completo, es difícil sacarle todo el partido que se podría, ya que posee su propia librería privada programada en C y su documentación es compleja, con lo que el tiempo de aprendizaje aumenta considerablemente. Además, cierta parte del código está reservada, lo que dificulta más su uso. Por esta razón se acabó descartando esta opción.

Más tarde, y una vez realizado el prototipo con Arduino presentado en las secciones anteriores, se comenzó la implementación de un prototipo más cercano a la solución propuesta. Para ello se utilizaron microcontroladores ATtiny85. Funcionan a

[26] https://www.silabs.com/products/development-tools/wireless/bluetooth/dkble-bluetooth-smart-development-kit

8MHz, con 512 bytes de RAM y 8 KB de memoria. Dadas las bajas prestaciones de este microcontrolador y su pequeño almacenamiento, hubo que adaptar el código utilizado, empeorando su calidad. A pesar de que la librería de comunicación serie es totalmente compatible y existen muchos ejemplos en los que la usan de forma efectiva, no se pudo obtener una comunicación satisfactoria con el módulo bluetooth. Se presupone que por razones de sincronización de las comunicaciones o de los retrasos introducidos en el código.

En consecuencia, se comenzó la implementación mediante el uso de dispositivos Digispark Pro[27] (Ilustración 24). Estos dispositivos contienen un microcontrolador ATtiny167 de 16MHz, 512 bytes de RAM y 16 KB de memoria. Para ello, hubo que soldar los conectores y adaptar también el código utilizado para este dispositivo. Finalmente, y a pesar de estar a punto de conseguirlo, no se pudo terminar el desarrollo con este dispositivo debido a un funcionamiento anormal a la hora de transmitir el carácter “+”, indispensable para la comunicación con el módulo bluetooth, en el que en su lugar transmite un salto de línea. También se ha probado el uso de otras librerías, pero el error persiste, lo cual puede significar que puede ser un error producido por el compilador. Una solución para obviar este problema puede ser usar módulos bluetooth que no usen comandos AT.

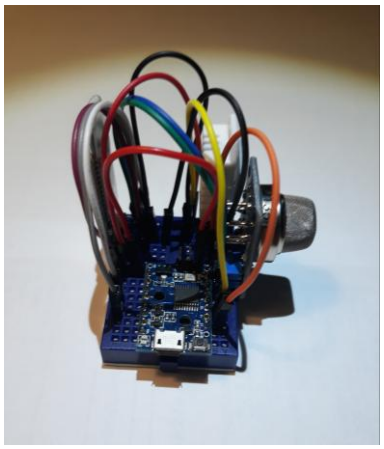

Ilustración 24. Foto del prototipo utilizando un Digispark Pro

[27] http://digistump.com/getpro

6. Plan de pruebas

En esta sección se describe un plan de pruebas que podría realizarse para analizar el sistema y comprobar su correcto funcionamiento. De esta forma se comprueba que los resultados obtenidos concuerdan con los esperados y, por ende, si el proyecto ha sido un éxito o no.

6.1. Planteamiento de las pruebas

Para el diseño del plan de pruebas, se plantea la posibilidad de realizar una prueba de 8 días de duración en un recinto cerrado en las afueras de la ciudad. De tal modo que dicho recinto o parcela disponga de suficiente espacio para situar los motes. La disposición de los motes deberá conformar, entre los 4, un rombo. Una vez situados los motes en sus respectivos lugares y en funcionamiento, se podrá proceder a la realización de las pruebas.

Las pruebas que se plantean son cuatro y se describen a continuación:

- Comprobación de la fiabilidad e integridad. Simplemente se deberá dejar el sistema funcionar durante 7 días para corroborar que no surge ningún problema o que algún mote deja de funcionar por cualquier causa.
- Comprobación de la tolerancia a fallos. Se procederá a la supresión o apagado de uno de los motes. De esta forma se podrá comprobar que el sistema se adapta a la situación y los datos llegan por otra ruta. Además, se deberá observar en la interfaz gráfica que el mote ha dejado de enviar datos.
- Comprobación de la escalabilidad y adaptabilidad. Mediante la activación de nuevos motes, éstos deben unirse a la red de forma automática y transmitir sus datos por ésta. Los nuevos datos deberán aparecer en la interfaz gráfica.
- Comprobación de la correctitud del sistema. Se procederá a hacer saltar la alarma en el sistema mediante el uso de alguna fuente de calor, como un secador o una cerilla. En la interfaz gráfica deberá aparecer la alarma detectada.

Si el sistema cumple satisfactoriamente las cuatro pruebas, se podrá decir que se cumplen las especificaciones descritas y el sistema se considerará funcional y correcto. En caso contrario, querrá decir que el sistema no es viable para su implantación, al menos por el momento.

6.2. Ejecución de las pruebas

Aunque finalmente las pruebas no se han realizado al aire libre en su totalidad, se han realizado en una vivienda, situando los motes en estancias separadas. La primera de las pruebas tenía una duración de 7 días, por lo que ha sido la más larga. En cambio, el resto de las pruebas se han podido realizar en un solo día sin que los resultados interfiriesen entre ellos.

Para la primera prueba, comprobación del funcionamiento continuo del sistema, no se ha observado ninguna incidencia en la ejecución del algoritmo que pudiese bloquearlo o saturarlo. Aunque no se mostraba la información del puerto serie de todos ellos, todos continuaban su ejecución sin problema, por lo que se considera que la prueba ha sido satisfactoria.

La segunda prueba, comprobación de la tolerancia a fallos, ha sido más sencilla. Para la prueba, primero se ha procedido a desconectar el mote situado en el lado derecho del rombo. Al cabo de unos minutos, el sistema ha avisado de la cesión del envío de datos de dicho mote. En cambio, se seguían recibiendo los datos del mote superior, lo cual significa que los datos se recibían por el mote situado a la izquierda del rombo que conforman. Después se ha vuelto a reactivar dicho nodo, apareciendo de nuevo en la interfaz. Seguidamente se ha procedido seguir el mismo procedimiento con el mote de la izquierda. Se seguían recibiendo los datos del mote superior, pero esta vez parece que el camino que seguían era por el mote de la derecha. De esto se concluye que la red se recupera muy rápidamente frente a fallos, por lo que la prueba se considera satisfactoria.

En la tercera prueba, comprobación de la escalabilidad y adaptabilidad, se ha partido de una red con 2 motes, a la cual se le han añadido 3 motes más activados a la misma vez, dos situados en el extremo superior del rombo y uno en el lado derecho del rombo. Dichos nuevos motes se han situado a cierta distancia para que no pudiesen enviar sus datos directamente al nodo servidor, forzándolos a que sus datos fuesen enrutados por la red. Como resultado, los datos han circulado correctamente por la red y han sido mostrados en la interfaz gráfica, obteniendo un resultado positivo en esta prueba.

La última prueba, comprobación de la detección de incendios, se ha realizado con cierta dificultad, ya que la prueba se ha realizado prendiendo cerillas dentro de un bote de aluminio, lo cual no lo convierte en las mejores condiciones para mantener una llama encendida mucho tiempo. De igual forma, la prueba se ha realizado con el mote situado en el extremo superior del rombo, así se fuerza a que los datos sean enrutados. Dado que la llama de la cerilla era escasa, no ha saltado la alarma por la temperatura. En cambio, sí que ha saltado por el índice de calidad del aire, ya que el humo que desprendía se expandía con facilidad. En un segundo intento, se ha realizado la prueba en el mismo mote mediante el uso de un secador. En esta ocasión,

la alarma sí que ha saltado por temperatura. Habiendo obtenido un resultado positivo en ambos intentos de esta prueba, puede darse como superada también.

A nivel general, y es algo que ya se conocía, se ha observado que la distancia de alcance de los dispositivos se ve mermada por los obstáculos que hay entre cada dispositivo. Hecho que hay que tener en cuenta a la hora de desplegar la red en una zona no despejada.

6.3. Resultados obtenidos

Tal como se puede apreciar en el apartado anterior, las pruebas muestran unos resultados satisfactorios, habiendo pasado las 4 pruebas sin complicaciones. Cabe señalar que las pruebas no se han realizado en las mismas condiciones en las que habría en un incendio, donde factores como el sol, el viento y la vegetación entre otras, puede influir en el resultado obtenido. De igual forma, se descubre y destaca que el uso de un sensor de índice de calidad del aire es muy útil para la detección de incendios, ya que permite su detección de una forma precoz, incluso cuando aún no se podría llamar de incendio forestal.

7. Conclusiones

En este capítulo se recogen las conclusiones del proyecto desarrollado, así como una valoración de los conocimientos obtenidos y la experiencia que ha conllevado su realización. También, se realiza un repaso a todos aquellos conocimientos que se han obtenido a lo largo de la carrera y que mediante este proyecto se han puesto en práctica.

7.1. Conclusiones del proyecto

En la realización de este proyecto, se ha analizado la problemática ocurrente sobre los incendios forestales, sus consecuencias tanto para el medio ambiente, como para la sociedad. Sus consecuencias en esta nuestra Comunidad Valenciana, en nuestro país. Y cómo IoT y las SmartCities pueden ayudar a luchar contra ellos.

Mediante el análisis y la comparación de las topologías de red y los algoritmos de enrutamiento, se ha aprendido que a veces, problemas complejos como el de este

proyecto, se pueden resolver con soluciones tan simples como el uso de un algoritmo de enrutamiento por inundación.

Indudablemente, mediante el estudio de la historia de las redes de sensores, se ha aprendido que fueron desarrolladas hace muchos años y que hoy en día se usan más de lo que percibimos en el día a día. Además, se ha analizado la estructura y los componentes de las redes de sensores y los dispositivos que las integran. A través el planteamiento de crear un mote personalizado, se ha aprendido que no se tratan de dispositivos complejos y que su desarrollo puede ser más rápido de lo que se espera. Junto con ello, también se han analizado propuestas comerciales de motes que, aunque son una solución rápida, no siempre son la mejor opción.

Para el estudio de la posible solución, se ha analizado un caso base y se han obtenido los requisitos que, más tarde, han servido para identificar las especificaciones que debía seguir el sistema. De esta forma, el problema ha sido más fácil de abordar y seleccionar una solución factible al problema planteado.

Finalmente, se ha desarrollado un prototipo que, aunque no se trata de los componentes propuestos, cumple con las exigencias del proyecto y se ha desarrollado una interfaz en el servidor para la presentación de los datos obtenidos por el sistema. Mediante una batería de pruebas, se ha podido comprobar que se ha conseguido desarrollar un sistema con unas altas capacidades de escalabilidad, autonomía y tolerancia a fallos, características fundamentales dadas las características del entorno donde se sitúan los sensores de la red.

Cabe destacar también que, mediante las pruebas realizadas, se ha podido observar que el índice de calidad del aire es un buen indicador de fuego, tanto como la temperatura y la humedad. Por lo que cabe destacar que su integración en los motes ofrece una ventaja que puede ser un factor clave para la mejora de la detección de incendios forestales.

En conclusión, se han obtenido los conocimientos y las destrezas necesarias para poder desarrollar, no solo una red de sensores, sino un algoritmo personalizado para ésta, que abarca las características del problema planteado en este proyecto y su finalidad intrínseca, acabar con los incendios forestales.

Tal como se ha dicho en puntos anteriores, en las redes de sensores da la sensación de que los datos se propagan de la misma forma que el fuego en un incendio. En ese caso, combatamos el fuego con fuego.

7.2. Relación con los estudios cursados

En este proyecto se han puesto en práctica diversos conocimientos que han sido aprendidos y desarrollados durante toda la carrera. Pudiendo distribuirlos por

materias podemos encontrar conocimientos de redes y programación, pero también conceptos básicos de ingeniería de software y de dispositivos de desarrollo de hardware. Por supuesto, no se han aplicado todos los conocimientos adquiridos durante mi formación académica universitaria. Pero otros conocimientos que se han ido adquiriendo de manera altruista o por motivación personal, sí se han aplicado en la realización de este proyecto. Incluso se han obtenido otros nuevos durante su realización, como el caso de la importancia de las redes de sensores en la actualidad y sus cientos de aplicaciones, que generan más interés en el tema. Así como una primera toma de contacto con el uso y programación de microcontroladores más robustos.

7.3. Apreciación personal

A modo de apreciación personal, es estupendo especializarse en una materia concreta, pero es esencial conocer también parte del resto de materias y así tener una visión global desde la que enfrentar cualquier reto, algo muy importante en el desarrollo de proyectos informáticos. La realización de este proyecto no habría sido posible si no hubiese sido ese el caso.

Aunque he obtenido una muy buena experiencia al realizar un proyecto personal que llevaba mucho tiempo queriendo realizar. Me ha sabido a poco ya que, por motivos laborales entre otros, no he podido asignarle todo el tiempo y constancia que me habría gustado, y poder desarrollar más a fondo la idea que he ido construyendo durante años.

La realización de este proyecto me ha hecho ver puntos muy interesantes del desarrollo de proyectos informáticos, que hasta ahora me parecían inalcanzables, y hasta qué punto pueden ser atractivos. Al principio, pueden parecer tareas tediosas, pero cuando las realizas por primera vez, te das cuenta de que están al alcance de todos, sólo es necesario un poquito de motivación y, sin duda, tiempo.

8. Futuras ampliaciones

Dados los positivos resultados del proyecto, ciertos cambios podrían mejorar varios aspectos de funcionamiento. Por ejemplo, añadir a la predicción de fuego la variable de nivel de CO, con lo que bastaría con añadir a los nodos un sensor de monóxido de carbono. Como ya se ha visto anteriormente, es uno de los principales gases que se emiten en la combustión, y esto permitiría a los nodos detectar con

mayor rapidez y precisión un incendio, además de ayudar a los servicios de emergencia a identificar en qué estado se encuentra el incendio.

Por otro lado, se han creado proyectos para la geolocalización de los nodos mediante triangulación con envíos de eco. Sería interesante aplicarle esta funcionalidad para así, geolocalizar los nodos sin intervención humana ni gasto energético de un localizador GPS.

Aunque sin duda, una de las posibles mejoras que se podría dar, sería dotar a los motes de un algoritmo para la gestión de la energía mediante la consulta de la previsión meteorológica. De forma que, dependiendo de la previsión meteorológica del día siguiente, aumente o disminuya los intervalos de actividad. Con ello, si la previsión meteorológica fuera de cielos cubiertos o lluvias, el nodo disminuiría el intervalo de actividad. Por el contrario, si la previsión es de un día soleado con un fuerte calor, el nodo incrementaría el intervalo de actividad, permitiendo una detección más temprana.

Glosario

BLE: Bluetooth Low Energy

BLR: Bluetooth Long Range

CO: Monóxido de carbono

CO_2: Dióxido de carbono

DARPA: Defense Advanced Research Projects Agency

DK: Development Kit

DSN: Distributed Sensor Network

IDE: Integrated Development Environment

IoT: Internet of Things

LED: Light Emitting Diode

MAC: Media Access Control

MBC: Meteor Burst Communication

PCB: Printed Circuit Board

RTC: Real Time Clock

SDK: Software Development Kit

SOSUS: Sound Surveillance System

UC: Unidad Computacional

VHF: Very High Frequency

WSN: Wireless Sensor Network

Wi-Fi: Wireless Fidelity

Bibliografía

[1] Jindal, V. (2018). History and Architecture of Wireless Sensor Networks for Ubiquitous Computing. *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)*, *7*(2).

[2] Iyengar, S. S. (Sundararaja S. ., & Brooks, R. R. (Richard R. . (2005). *Distributed sensor networks*. Chapman & Hall/CRC.

[3] Ng, J. W. P., Ng, J. W. P., Lo, B. P. L., Wells, O., Sloman, M., Peters, N., … Yang, G. (2004). Ubiquitous Monitoring Environment for Wearable and Implantable Sensors (UbiMon).

[4] Sanchez, L., Muñoz, L., Galache, J. A., Sotres, P., Santana, J. R., Gutierrez, V., … Pfisterer, D. (2014). SmartSantander: IoT experimentation over a smart city testbed. *Computer Networks*, *61*, 217-238. https://doi.org/10.1016/j.bjp.2013.12.020

[5] McKinsey Global Institute. (2018). SMART CITIES : DIGITAL SOLUTIONS FOR A MORE LIVABLE FUTURE, (June).

[6]Kurose, J. F. (2010). *Redes de computadoras: un enfoque descendente*. (K. W. Ross, Ed.) (Trad. de la 5ª ed. en inglés.). Madrid: Madrid : Prentice Hall, D.L. 2010.

[7] Woolley, M. (s. f.). Introducing Bluetooth Mesh Networking | Bluetooth Technology Website. Recuperado 26 de noviembre de 2018, de

[8] Hart, A. R. D. (1985). *Incendios Forestales*. *Sistemas Ecológicos*. Universidad Autónoma Chapingo. https://doi.org/10.1109/TAEECE.2015.7113639

[9] Ministerio de Agricultura, A. y M. A. (s. f.). Estadísticas de Incendios Forestales 2018. Recuperado 20 de noviembre de 2018, de https://www.mapa.gob.es/es/desarrollo-rural/estadisticas/Incendios_default.aspx

[10] Ministerio de Agricultura, A. y M. A. (s. f.). Estadísticas de Incendios Forestales hasta 2010. Recuperado 20 de noviembre de 2018, de https://www.mapa.gob.es/es/desarrollo-rural/temas/politica-forestal/incendios-forestales/default_estadisticas_forestales.aspx

Ilustración 1. Ventajas de las Smart Cities: (http://smartsocialcity.com/wp-content/uploads/2015/01/Mapa-13-%C3%81reas-1024x576.png)

Anexo 1. Código Arduino del algoritmo

```
#include<CircularBuffer.h>
#include<SoftwareSerial.h>
#include <DHT.h>

/*
 * Autor: Rubén Quintanilla Lahiguera
 * Año: 2019
 * Proyecto: Diseño e implementación de una red inalámbrica de sensores para la detección precoz
 *           de incendios forestales.
 * Módulo: Mote
 * Descripción: El mote espera conexiones entrantes para reenviarlas a sus dispositivos vecinos.
 *             También es capaz de detectar réplicas y descartarlas, y evitar bucles de inundación.
 *             tiempo aleatorio inicia el envío de sus propios datos de temperatura, humedad y
 *             calidad del aire a todos los dispositivos cercanos.
 */

//Definición de los pines necesarios
#define DHTTYPE DHT22     //Tipo de sensor de temperatura/humedad.
#define DHTPIN 5    //Pin del sensor de temperatura/humedad.
#define type 135    //Tipo del sensor de calidad de aire (ICA).
#define aqPin A0    //Pin del sensor ICA.

//Pines del bluetooth.
const int TXPin = 11;
const int RXPin = 12;
const int BRKPin = 2;

SoftwareSerial BT(TXPin,RXPin);    // Definimos los pines RX y TX del Arduino conectados al Bluetooth.
DHT dht(DHTPIN, DHTTYPE);

//Inicialización de variables globales.
const float alertaTemperatura = 50.0;
const int alertaICA = 350;
const int numIteracionesNormal = 99;
const int numIteracionesAlerta = 50;

String id = "";                 //MAC.
const String latitud = "+397127";   //Latitud
const String longitud = "-0004631";  //Longitud
float temperatura = 0;                    //Temperatura
float humedad = 0;                   //Humedad
float ica = 0;                //ICA

CircularBuffer<String, 8> checksums;
String nodoAnt = " ";

boolean alertaIniciada = false;
int iteraciones;

void setup() {
  BT.begin(9600);   //Inicialización del puerto serie BT.
  Serial.begin(9600);   //Inicialización del puerto serie.
  dht.begin();    //Inicialización del sensor DHT22.
  pinMode(BRKPin, OUTPUT);
  delay(1000);
  iteraciones = numIteracionesNormal;

  configurar();
  enviarComando("AT+ROLE0");    //Pone el bluetooth en modo periférico para recibir conexiones.
  enviarComando("AT+START");    //Inicia la recepción de conexiones.
}

void loop() {
  //Bucle para comprobar la recepción de datos.
  //Define la iteración del bucle en que realizará el envío de datos mediante la generación de
  //un número aleatorio.
  int tDifusion = random(0, iteraciones);
  //Interrumpe las conexiones activas del bluetooth si las hubiera.
```

```
  digitalWrite(BRKPin, LOW);
  delay(1000);
  digitalWrite(BRKPin, HIGH);
  Serial.println("Abierto a conexión...");

  //Bucle de recepción/envío.
  for (int i = 0; i < iteraciones + 1; i++) {
    //Si el número aleatorio coincide con la iteración del bucle, difunde sus propios datos.
    if (i == tDifusion) {
      difundir(generarCadena());
      Serial.println("Abierto a conexión...");
    } else {
      delay(200);
      //Si hay datos disponibles para recibir, los lee. Si no, toma datos de los sensores.
      if (BT.available()) {
        String resp = BT.readString().c_str();
        //Si el dato leído es una coexión establecida, ejecuta el método recibir.
        if (resp.indexOf("OK+CONN") != -1) {
          recibir();
          Serial.println("Abierto a conexión...");
        }
      } else {
        //Toma las mediciones de los sensores.
        temperatura = dht.readTemperature();
        humedad = dht.readHumidity();
        ica = analogRead(aqPin);
        //Establece el máximo del nivel ICA.
        if (ica > 9999) {
          ica = 9999;
        }
        //Si en esta iteración detecta un peligro, sitúa el bucle en la iteración anterior a
        // la del envío de datos y reajusta el número de iteraciones del bucle.
        if (!alertaIniciada && (temperatura > alertaTemperatura || ica > alertaICA)) {
          iteraciones = numIteracionesAlerta;
          i = tDifusion - 1;  //Iteración anterior a la del envío de datos.
          alertaIniciada = true;
        //Si ya no detecta un peligro, reestablece los ajustes del bucle al estado normal.
        } else if (alertaIniciada && (temperatura < alertaTemperatura && ica < alertaICA)) {
          iteraciones = numIteracionesNormal;
          alertaIniciada = false;
        }
      }
    }
  }
}

//Inicia la rutina de recepción de mensajes por bluetooth.
//Al recibir un mensaje, se evalúa si puede ser una réplica. En caso de serlo, lo descarta.
//Si no, lo difunde.
void recibir() {
  Serial.println("Recibiendo datos");
  String recibido = BT.readString();
  //Bucle para la recepción de mensajes
  //Se establece un bucle porque pueden haber retrasos y necesitar mas tiempo para recibir
  // los mensajes. También se establece un límite máximo para que no entre en un bucle
  // infinito si no le envían más mensajes.
  int f = 0;
  while (recibido.equals("") && f < 10) {
    recibido = BT.readString();
    //Descarta los mensajes que no sean cadenas.
    if (recibido.indexOf("AT+") != -1 || recibido.indexOf("OK+") != -1) {
      recibido = "";
    }
    f++;
    delay(200);
  }
  Serial.println("Recibido: " + recibido);
  String checksum = recibido.substring(10, 14);    //Extrae el número de secuencia.
```

```
   String nodoOri = recibido.substring(0, 12);      //Extrae la MAC origen.
   //Comprueba si la cadena ya ha sido recibida anteriormente.
   boolean esReplica = false;
   for (int i = checksums.size() - 1; i >= 0; i--) {
    esReplica = esReplica || checksum == checksums[i];
   }
   //Si la cadena no es una réplica, guarda el número de secuencia e inicia el método
   // difundir con la cadena recibida.
   if (!esReplica) {
    checksums.push(checksum);
    Serial.println(checksums[0]);
    Serial.println(checksums[1]);
    Serial.println(checksums[2]);
    Serial.println(checksums[3]);
    Serial.println(checksums[4]);
    Serial.println(checksums[5]);
    Serial.println(checksums[6]);
    Serial.println(checksums[7]);
    BT.readString();
    difundir(recibido);
   } else {
    Serial.println("Es una réplica");
   }
}

//Genera la cadena a enviar con los datos en el formato y posición definida.
//(MAC)(Número secuencia)Y(Latitud)X(Longitud)T(Temperatura)H(Humedad)Q(ICA)
String generarCadena() {
  //Genera el número de secuencia para esta cadena.
  String checksum = String(random(0, 9)) + String(random(0, 9));
  //4 digitos para la temperatura con ceros a la izquierda.
  char temString[5]; sprintf(temString, "%04d", int(temperatura * 10.0));
  //4 digitos para la humedad con ceros a la izquierda.
  char humString[5]; sprintf(humString, "%04d", int (humedad * 10.0));
  //4 digitos para el ICA con ceros a la izquierda.
  char icaString[5]; sprintf(icaString, "%04d", int(ica));
  return id + checksum + "Y" + latitud + "X" + longitud + "T" + String(temString) + "H" +  String(humString) +
"Q" + String(icaString);
}

//Difunde el parámetro dado a todos los dispositivos con nombre firesensor.
void difundir(String cadena) {
  //Interrumpe las conexiones activas del bluetooth si las hubiera.
  digitalWrite(BRKPin, LOW);
  delay(500);
  digitalWrite(BRKPin, HIGH);
  delay(500);
  Serial.println("Preparando cadena: ");
  Serial.println(cadena);

  CircularBuffer<String, 5> vecinos;
  enviarComando("AT+ROLE1");    //Pone el bluetooth en modo central para crear conexiones.
  BT.write("AT+DISC?");   //Inicia la búsqueda de dispositivos bluetooth.
 Serial.println("Buscando...");
  //Bucle para la recepción de los dispositivos encontrados. Se ejecuta por cada
  // dispositivo encontrado.
  int f = 0;
  while(f < 20) {
    if (BT.available()) {
      String resp = BT.readStringUntil('\n').c_str();
      //Comprueba si el dispositivo tiene el nombre firesensor.
      if (resp.indexOf("firesensor") != -1) {
        Serial.println(resp);
        //Extrae la dirección MAC del dispositivo al que conectarse.
        int inicio = resp.indexOf("DIS0:");
        String mac = resp.substring(inicio + 5, inicio + 17);
        //Evita que la cadena vuelva por el nodo que la envió.
        //Si la MAC de destino no es la misma que la de origen de la cadena ni el nodo
```

```
        // anterior, añade la MAC al listado de enviar.
        if (cadena.indexOf(mac) == -1 && cadena.indexOf(nodoAnt) == -1) {
          Serial.println("No es el origen ni el anterior.");
          //Previene que no sean demasiados vecinos a los que enviar la cadena.
          if (!vecinos.isFull()) {
            vecinos.push(mac);
           Serial.println (vecinos.last());
          }
        } else {
          Serial.println(mac);
          Serial.println(nodoAnt);
          Serial.println(resp);
          Serial.println("Es el origen o el anterior.");
        }
      //Si llega el mensaje de fin de dispositivos encontrados, sale del bucle.
      } else if (resp.indexOf("OK+DISCE") != -1) {
        Serial.println(resp);
        Serial.println("Ha salido");
        break;
      } else if (resp.indexOf("OK+CONN") != -1) {
        recibir();
      }
    } else {
      f++;
      delay(50);
    }
  }
  delay(2000);

  //Bucle para realizar la conexión con los dispositivos encontrados y enviarles la cadena.
  for (int j = 0; j < vecinos.size(); j++) {
    //Envia la orden de crear conexion al bluetooth.
    String resp = enviarComando("AT+CON" + vecinos[j]);
    delay(1000);
    //Si la conexión es satisfactoria, envia la cadena. Si no, descarta la conexión.
    if (resp.indexOf("OK+CONNA") != -1) {
      Serial.println("Conectado.");
      BT.write(cadena.c_str());
      Serial.println(cadena.c_str());
      Serial.println("Datos enviados");
    } else if (resp.indexOf("OK+CONNE") != -1) {
      Serial.println("Error en la conexión.");
    } else {
      Serial.println(resp);
    }
    delay(1000);
    //Cierra la conexión establecida con el nodo.
    digitalWrite(BRKPin, LOW);
    delay(1000);
    digitalWrite(BRKPin, HIGH);
    delay(1000);
    resp = BT.readString().c_str();
    if (resp.indexOf("OK+LOST")) {
      Serial.println("Desconectado.");
    }

  }
  //Resetea los valores de las variables globales.
  vecinos.clear();
  nodoAnt = " ";

  enviarComando("AT+ROLE0");    //Pone el bluetooth en modo periférico para recibir conexiones.
  enviarComando("AT+START");    //Inicia la recepción de conexiones.
}

//Configura inicialmente el modulo bluetooth.
void configurar() {
  //Interrumpe las conexiones activas del bluetooth si las hubiera.
```

```
  digitalWrite(BRKPin, LOW);
  delay(1000);
  digitalWrite(BRKPin, HIGH);
  enviarComando("AT+RENEW");          //Resetea la configuración del bluetooth a los valores de fábrica.

  enviarComando("AT+IMME1");          //Establece que el bluetooth no realice conexiones de forma automática.
  enviarComando("AT+PWRM1");          //Establece que el dispositivo no pase a modo Sleep de forma automática.
  enviarComando("AT+PASS000000");     //Establece la contraseña de conexión.
  enviarComando("AT+NAMEfiresensor"); //Establece el nombre del bluetooth.
  enviarComando("AT+NOTI1");          //Establece que el bluetooth envíe un mensajes de estado de conexión.
  enviarComando("AT+TYPE0");          //Establece que no se solicite pin de conexión.
  enviarComando("AT+MODE0");          //Establece que NO se puedan ejecutar comandos de forma remota.
  enviarComando("AT+ROLE1");          //Pone el bluetooth en modo central para crear conexiones.
  enviarComando("AT+FILT0");          //Establece que se pueda conectar a cualquier bluetooth.
  enviarComando("AT+SHOW1");          //Establece que el bluetooth sea visible para otros dispositivos.
  enviarComando("AT+ADTY0");          //Establece que el bluetooth pueda ser conectable por otros dispositivos.
  delay(500);
  id = enviarComando("AT+ADDR?").substring(8, 21);    //Consulta y almacena la MAC de este bluetooth.
  Serial.println(id);

  enviarComando("AT+RESET");    //Resetea el bluetooth.
  //Genera la semilla para crear números aleatorios.
  //Dado que Arduino siempre crea la misma secuencia de números aleatorios
  // con una misma semilla,se le da como semilla el valor analógico del pin A0,
  // que será un dato de ruido, y por lo tanto aleatorio, al no haber nada conectado.
  randomSeed(analogRead(A0));
  delay(250);
}

//Envía un comando dado al bluetooth y devuelve su respuesta.
String enviarComando(String comando) {
  BT.write(comando.c_str());
  Serial.println(comando);
  String respuesta = "";
  int i = 0;
  //Se realiza con un bucle finito porque la respuesta puede no llegar instantánea.
  while (i < 10) {
    if(BT.available()) {
      respuesta = respuesta + BT.readString().c_str();
      break;
    }
    delay(100);
    i++;
  }
  Serial.println("-> " + respuesta);
  return respuesta;
}
```