Universitat Oberta de Catalunya

uoc.edu

# Diseño e implementación de un sistema de emergencias doméstico global basado en redes de sensores inalámbricas

**Eduardo Pérez Galán**
Máster Universitario en Ingeniería de telecomunicación
Sistemas de comunicación

**Profesor colaborador:**
**Raúl Parada Medina**

**Profesor responsable de la asignatura:**
**Carlos Monzo Sánchez**

21 de junio de 2020

Esta obra está sujeta a una licencia de Reconocimiento-NoComercial-SinObraDerivada 3.0 España de Creative Commons

**FICHA DEL TRABAJO FINAL**

| | |
|---|---|
| **Título del trabajo:** | *Diseño e implementación de un sistema de emergencias doméstico global basado en redes de sensores inalámbricas* |
| **Nombre del autor:** | *Eduardo Pérez Galán* |
| **Nombre del consultor/a:** | *Raúl Parada Medina* |
| **Nombre del PRA:** | *Carlos Monzo Sánchez* |
| **Fecha de entrega (mm/aaaa):** | 06/2020 |
| **Titulación:** | *Máster universitario en ingeniería de telecomunicación* |
| **Área del Trabajo Final:** | *Sistemas de comunicación* |
| **Idioma del trabajo:** | *Español* |
| **Palabras clave** | *LPWAN, WPAN, Redes de sensores, Sistema de emergencias, IoT, detección, incendio.* |

**Resumen del Trabajo (máximo 250 palabras):** *Con la finalidad, contexto de aplicación, metodología, resultados i conclusiones del trabajo.*

Dentro de los sistemas de comunicaciones, el campo de estudio orientado a IoT es uno de los que mayor progresión ha experimentado. La posibilidad de monitorizar diferentes parámetros a través de redes formadas por miles de nodos-sensores permite una reducción de los costes producidos tanto a nivel industrial como doméstico, así como un menor impacto en el medio ambiente.
Ante esta mejora, la tecnología asociada a esta área ha evolucionado rápidamente ofreciendo sistemas con coberturas punto-punto del orden de kilómetros o consumos tan bajos que permiten una autonomía de los nodos durante años sin alimentación externa.

El presente trabajo quiere analizar, diseñar e implementar, a todos los niveles, una red de sensores conectada mediante diferentes sistemas de comunicación tan diferentes como lo son los sistemas WPAN y LPWAN. El objetivo principal será la detección de posibles emergencias a través de la medición de diferentes parámetros por parte de cada nodo. Si alguno o varios parámetros superan el umbral de detección, se transmitirá una alarma por la red hasta el destino.

Aprovechando las ventajas de estos sistemas de comunicación, se quiere conseguir un sistema de detección fiable, de bajo consumo y con una gran cobertura de acción a un coste inicial de infraestructura y mantenimiento bajo. Además, el modelo final de nodo-sensor, estará diseñado de forma modular,

permitiendo una actualización continua de los componentes, o la adición posibles mejoras como la incorporación de sistemas de recolección energética, lo que le proporcionaría una autonomía indefinida permitiéndole su posicionamiento en lugares desatendidos.

**Abstract (in English, 250 words or less):**

Within communication systems, the field of study oriented to IoT is one of those that has experienced the greatest progression. The possibility of monitoring different parameters through networks made up of thousands of sensor-nodes allows a reduction in costs cost reduction in both industrial and domestic levels, as well as a lower impact on the environment. Considering this improvement, the technology associated to this area has quickly evolved offering systems with point-to-point coverage on the order of kilometers or consumption so low that they allow autonomy of the nodes for years without external power.

The present document aims to analyze, design and implement, at all levels, a sensor network connected by different communication systems diverse from WPAN and LPWAN systems. El main purpose will be the detection of potential emergencies through the measurement of different parameters by each node. If one or more parameters exceed the detection threshold, an alarm will be transmitted through the network to the destination.

Taking advantage of these communication systems, we want to achieve a reliable detection system, with a lower power consumption and a large coverage of action besides a low infrastructure and maintenance cost. In addition, the final model of sensor will be modular block designed, allowing a continuous component updating or the addition of possible improvements such as the incorporation of energy harvesting systems, which would provide an extended autonomy.

# Índice


1. Introducción ..... 1
1.1 Contexto y justificación del Trabajo ..... 1
1.2 Objetivos del Trabajo ..... 3
1.3 Enfoque y método seguido ..... 4
1.4 Planificación del Trabajo ..... 5
1.5 Breve sumario de productos obtenidos ..... 8
1.6 Breve descripción de los otros capítulos de la memoria ..... 9
2. Estado del arte ..... 10
2.1 Generalidades ..... 10
2.1.1 Concepto de emergencia ..... 10
2.1.2 Sistemas detección de incendios ..... 10
2.2 Métodos de detección de incendios. Sensores. ..... 12
2.2.1 Detectores iónicos de humos [11] ..... 13
2.2.2 Detectores de humos visible o fotoeléctricos ..... 13
2.2.3 Detectores térmicos [12]. ..... 14
2.2.4 Detectores de radiación. ..... 14
2.3 Comunicaciones inalámbricas orientadas a comunicación de sensores. 14
2.3.1 Comunicaciones WPAN ..... 15
2.3.2 Comunicaciones LPWAN ..... 16
2.4 Ejemplos de aplicación para redes de sensores mixtos ..... 17
2.4.1 Ejemplo 1: Low Energy Smart Cities Network with LoRa and Bluetooth ..... 17
2.4.2 Ejemplo 2: Fire Detection. Semtech ..... 19
2.4.3 Ejemplo 3: Solución comercial Siterwell ..... 20
2.4.4 Ejemplo 4: Mobile Fire alarm using WSN (*Wireless Sensor Network*). ..... 20
2.4.5 Ejemplo 5: Alarma de seguridad comercial para uso domestico ..... 21
2.5 Contribución del trabajo al area de estudio. ..... 23
3. Fundamentos teóricos de sistema de redes de sensores. ..... 27
3.1 Introducción ..... 27
3.2 Características ..... 27
3.3 Arquitectura de un nodo/sensor. ..... 29
4. Desarrollo de PCB modular con conectividad BLE, WiFi y LoRa. ..... 31
4.1 Sistema modular. Descripción y especificaciones técnicas. ..... 31
4.2 Esquema ..... 38
4.3 Rutado ..... 39
4.4 Documentación/fabricación ..... 44
4.4.1 Documentación para fabricación de PCB ..... 44
4.4.2 Documentación para montaje de componentes en PCB ..... 45
4.4.3 Otros documentos ..... 46
4.5 Montaje y puesta en marcha ..... 47
4.6 Programación e incorporación de los prototipos al sistema. ..... 49
5. Desarrollo de sistema de emergencias ..... 50
5.1 Introducción a la arquitectura del sistema emergencias ..... 50
5.2 Elementos del sistema ..... 53

5.3 Implementación de arquitectura. ..... 54
5.3.1 Fase 1: Lectura de parámetros de medida en sensores y generadores de alarma en el nodo. ..... 55
5.3.2 Fase 2: Comunicación WPAN BLE nodos-gateway..... 58
5.3.3 Fase 3: Comunicación Gateway-Backend Sigfox ..... 62
5.3.4 Fase 4: Gestión de Callback y mensajes de dispositivos en la nube Sigfox..... 65
5.3.5 Fase 5: Servidor de cliente, almacenamiento y visualización de alarmas. ..... 66
6. Pruebas sobre el sistema completo y análisis de los resultados. ..... 70
6.1 Configuración de sistema para pruebas. ..... 70
6.2 Primeras pruebas sobre sistema cableado. ..... 72
6.3 Pruebas del sistema completo ..... 74
6.4 Pruebas de medición de temperatura..... 75
6.5 Pruebas de medición de gases. ..... 76
6.6 Pruebas de consumo..... 77
6.7 Análisis de resultados..... 79
7. Valoración económica del trabajo ..... 80
7.1 Costes asociados al desarrollo del prototipo. ..... 80
7.2 Coste asociado a Ingeniería. ..... 82
7.3 Coste total de proyecto..... 83
8. Conclusiones..... 84
9. Glosario ..... 89
10. Bibliografía ..... 91
11. Anexos ..... 98

## Lista de figuras

*Figura 1: Relación entre tamaño del incendio o probabilidad de muerte con el tiempo de respuesta de los servicios de emergencias* [3]. ..... 1
*Figura 2: Evolución de víctimas mortales en España hasta 2018* [4] ..... 2
*Figura 3: Componentes de un sistema de detección automática de incendios [7]* ..... 11
*Figura 4: Fases de actuación de los detectores.[8]* ..... 12
*Figura 5: Funcionamiento de un detector iónico de humos [9]* ..... 13
*Figura 6: Funcionamiento de un detector fotoeléctrico [9]* ..... 13
*Figura 7: Arquitectura LoRa + BLE [21]* ..... 18
*Figura 8: Propuesta sistema de detección de incendios a través de LoRa [22]* 19
*Figura 9: Arquitectura de sistema detección de incendios por Siterwell* [23] .... 20
*Figura 10: Arquitectura de detección basada en conectividad WiFi* [24] ..... 21
*Figura 11: Diagrama de sistema de seguridad doméstica basada en GSM/GPRS/LTE* [25] ..... 22
*Figura 12: Principales aportaciones de cada sistema de comunicación a la arquitectura.* ..... 23
*Figura 13: Diagrama genérico de la arquitectura de un nodo/sensor [35]* ..... 29
*Figura 14: Diagrama de sistema electrónico modular ESP32* ..... 32
*Figura 15: modulo micro-32 TTGO/diagrama interno de ESP32* ..... 33
*Figura 16: Modulo transceptor ISM 868 MHz basado en chip Semtech SX1276* [42] ..... 35
*Figura 17: Encapsulado de sensor Bosch BME680* [45] ..... 36
*Figura 18: Encapsulado típico de Buzzer de inserción* ..... 37
*Figura 19: Ejemplo de prepreg comerciales [54].* ..... 41
*Figura 20: Stackup de diseño modular para 4 capas y 1,6 mm de grosor, JLCPCB [55].* ..... 42
*Figura 21: Captura de distribución de antena cerámica y ULF para conectividad LoRa* ..... 43
*Figura 22: Ajuste de impedancia para entrada de antena LoRa.* ..... 43
*Figura 23: Capa TOP de diseño PCB modular* ..... 44
*Figura 24: Representación 3D de PCB modular de nodo/sensor.* ..... 46
*Figura 25: PCB diseñada en este trabajo.* ..... 47
*Figura 26: PCB principal y módulos accesorios montados y verificados eléctricamente.* ..... 47
*Figura 27: Dispositivos preparados para su puesta en marcha y posterior integración con mecánica y batería.* ..... 48
*Figura 28: Diagrama de dispositivos de la arquitectura.* ..... 49
*Figura 29: Capa 1. Red WPAN para vivienda. plano de una vivienda.* ..... 50
*Figura 30: Representación de Capa 2 y 3 de arquitectura.* ..... 51
*Figura 31: Visión global de sistema de emergencias.* ..... 52
*Figura 32: Prototipo de nodo/sensor a partir de placa genérica + BME680+ altavoz + Batería* ..... 53
*Figura 33: Prototipo gateway LoRa a partir de placa TTgo T3 + Batería* ..... 54
*Figura 34: Gateway Sigfox. Lopy4 de Pycom + Placa de expansión + Batería* 54
*Figura 35: Diagrama de flujos de firmware nodo/sensor* ..... 55
*Figura 36: Modificaciones en placa para trabajar en modo Deep sleep.* ..... 56

*Figura 37: Mecanismos de interacción en la comunicación entre dispositivos dentro de una red BLE.* ..... 59
*Figura 38: Representación de un perfil BLE.* ..... 60
*Figura 39: Diagrama de flujos de firmware de gateway LoRa* ..... 61
*Figura 40: Estructura de protocolo LoRa RAW.* ..... 62
*Figura 41: Diagrama de flujo de firmware de Gateway Lora-Sigfox en Lopy4 de Pycom* ..... 64
*Figura 42: Panel de control web para configuración de dispositivos y datos asociados Sigfox.* ..... 65
*Figura 43: Diagrama de sistema de emergencias basado en servidor virtual.* . 67
*Figura 44: Panel de control de sistema de detección de emergencias.* ..... 69
*Figura 45: Diagrama de arquitectura propuesta para pruebas reales* ..... 70
*Figura 46: Elementos utilizados para la prueba del sistema* ..... 71
*Figura 47: Arranque de red WPAN, visualización de consola por puerto serie* 72
*Figura 48: Escaneo de dispositivos por parte del gateway LoRa y proceso de conexión.* ..... 73
*Figura 49: Proceso de envío de alarma. Parte 1. Dispositivos del Edificio*..... 74
*Figura 50*: *Proceso de envío de alarma. Parte 2. Recepción en servidor* ..... 74
*Figura 51: Proceso de envío de alarma. Parte 3. Listado de alarmas almacenadas* ..... 75
*Figura 52: Comparativa medición de gases. Medición factor IAQ entre dos nodos.* ..... 76
*Figura 53: Diagrama de medición de consumo sobre dispositivos*..... 77
*Figura 54: Pruebas de consumo sobre nodo/sensor*..... 78

## Lista de Tablas

*Tabla 1: planificación de tareas y recursos* ..... 6
*Tabla 2: Diagrama de Gantt* ..... 7
*Tabla 3: Comparativa de tecnologías usadas en sistemas analizados* [29][30][31][32] ..... 26
*Tabla 4: Modos de funcionamiento ESP32 y consumos.* ..... 34
*Tabla 5: Características más importantes de BME680* [45]. ..... 36
*Tabla 6: Tabla cuantificativa de parámetro IAQ con el sensor BME680* [45]. .. 36
*Tabla 7: Comparativa de consumo y autonomía de nodo según parámetros de medida y fuente de alimentación.* ..... 57
*Tabla 8: Comparativa de medida de gases entre diferentes sensores* ..... 77
*Tabla 9: Consumo de dispositivos en situaciones de comunicación y medida.* 78
*Tabla 10: Coste de lista de materiales de prototipos desarrollados.* ..... 81
*Tabla 11:Coste total de sistema prototipo para el trabajo* ..... 82
*Tabla 12: Costes asociados a gastos de ingeniería.* ..... 82
*Tabla 13: Coste total de trabajo* ..... 83

# 1. Introducción

## 1.1 Contexto y justificación del Trabajo

Cualquier vivienda o inmueble, así como las personas que lo habitan están sometidos a diferentes peligros provocados por factores externos. Estas amenazas, tales como fallos eléctricos, averías en las instalaciones como calderas de gas, cocina, electrodomésticos o incluso un error humano pueden poner en peligro la integridad y la vida de los habitantes. La cantidad de daño provocado dependerá directamente del tiempo de reacción y, por lo tanto, este factor es clave a la hora de analizar un sistema de estas características.
En la figura 1 (imagen inferior) se puede observar que la probabilidad de muerte no se incrementa de forma lineal, existe un punto en el que la probabilidad se asciende rápidamente debido al punto de propagación del incendio

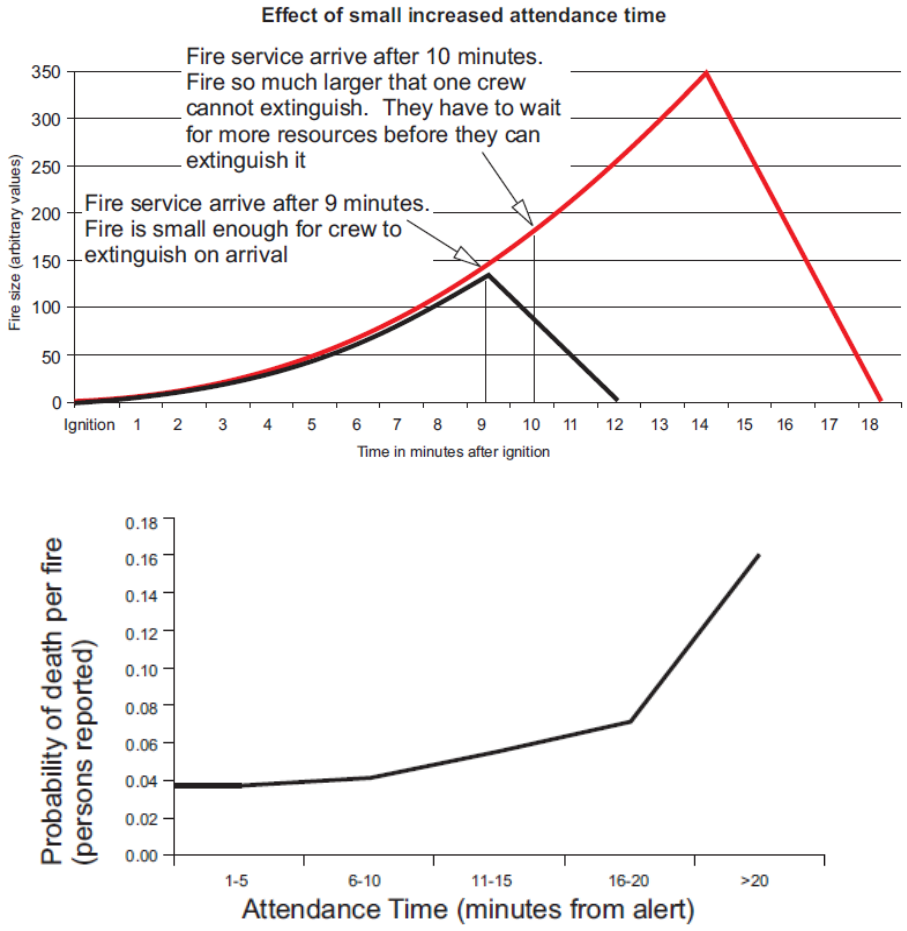



*Figura 1: Relación entre tamaño del incendio o probabilidad de muerte con el tiempo de respuesta de los servicios de emergencias* [3].

En la actualidad, la mayoría de los sistemas actuales de detección de incendios, calidad del aire y gases se trata de simples elementos sensores que mediante señales acústicas [1] avisan al usuario de la emergencia en cuestión. (Otros sistemas, pertenecientes a compañías privadas, disponen de conectividad con

una central, pero están más orientados a la vigilancia y a través de centralitas privadas [2]). Pero aun así en muchos casos, los propios individuos no se encuentran en un estado totalmente consciente y no pueden reaccionar de una forma apropiada a la emergencia, como es, por ejemplo, realizar una llamada o pedir auxilio, lo que hace que el tiempo de respuesta sea mucho mayor [3]. Por ello, surge la necesidad de un sistema automático y global que pueda controlar y avisar de forma automática a los centros responsables de atención de emergencias.

*Figura 2: Evolución de víctimas mortales en España hasta 2018* [4]
*Se puede observar un valor constante de muertos a lo largo de los años, eso sin incluir muertes indirectas por una lenta respuesta o violencia de genero.*

Por otro lado, respecto a los sistemas de comunicación inalámbricos y el concepto de IoT (*internet of things*), ambos han tomado una gran importancia y relevancia en los últimos años, tanto en el sector industrial como doméstico y han permitido grandes ventajas [5]:

- Un incremento en la eficiencia energética de las instalaciones.
- Una mayor productividad y por lo tanto un mayor beneficio, gracias a la monitorización de la maquinaria de la cadena de producción, reduciendo su tiempo de inactividad.
- Mejora continua de negocio. La monitorización de todos los aspectos cotidianos dentro de una empresa o domicilio permite la adquisición de grandes cantidades de datos. Un estudio de estos permite un mayor análisis de posibles estrategias para optimizar cualquier proceso. Desde un proceso de fabricación hasta la climatización de una vivienda.

A pesar de los progresos obtenidos, todavía queda mucho camino por recorrer en el campo de las redes de sensores inalámbricas. Por un lado, se puede encontrar como primera barrera, el coste económico de la infraestructura a

implantar, un ejemplo puede ser el uso de nodos con conectividad propietaria como es Sigfox. En este tipo de comunicaciones es necesario pagar por el uso de sus infraestructuras. Otro ejemplo es la tecnología LoRaWAN (*Long Range Wide Area Network*) que no dispone de una infraestructura propietaria, y por lo tanto es necesaria la inversión inicial para la creación de un sistema de estaciones para dar cobertura total a un área metropolitana.

Por lo tanto, si miramos ambas áreas de estudio, sistemas de emergencias y comunicaciones inalámbricas, es justificable el uso de una combinación de sistemas de comunicación inalámbricos y aprovechar las ventajas de cada uno de ellos para poder mejorar el tiempo de respuesta ante una emergencia intentando reducir al máximo los costes, pero manteniendo los objetivos principales del trabajo: robustez, bajo consumo, amplia cobertura y bajo coste.

## 1.2 Objetivos del Trabajo

Los principales objetivos del presente trabajo se pueden dividir en dos bloques:

- El primer objetivo se puede considerar como un estudio, análisis y comparativa de las tecnologías tanto en el campo de la detección de emergencias, como de las tecnologías actuales en sistemas de comunicación inalámbricos.

- El segundo objetivo se trata del desarrollo de una solución completa de un sistema de emergencias con el uso de las tecnologías seleccionadas previamente. Esta segunda tarea se puede dividir en dos subtareas a completar:

1- Desarrollo de un sistema de emergencias con tecnologías basadas en sistemas de comunicaciones inalámbricos orientados a IoT, este sistema debe cumplir cuatro condiciones:
    - Debe ser un sistema robusto
    - Disponer de una gran cobertura
    - Tener un consumo eléctrico muy bajo.
    - El coste debe ser reducido.

    Debido a la complejidad del sistema, se ha decidido realizar primero una prueba de concepto con módulos comerciales para comprobar que el planteamiento es correcto.

    El desarrollo del sistema se dividirá en 6 fases secuenciales:
    - Fase 1: Configuración de nodos, lectura de parámetros desde los módulos sensores de temperatura, calidad de aire y gases.
    - Fase 2: Desarrollo de firmware para comunicación WPAN (*Wireless personal area network)* entre nodos y Gateway con conectividad SigFox
    - Fase 3: Comunicación Gateway Sigfox con backend Sigfox
    - Fase 4: Configuración de servidor de datos y creación de base de datos con cruce entre ID nodo y dirección de vivienda.
    - Fase 5: Diseño de aplicación para la visualización de emergencias

- Fase 6: Tareas de optimización
    1. Cambio de topología de red WPAN a topología Mesh
    2. Incorporación capa de comunicación LPWAN (*Low Power Wide Area Network*).

2- Desarrollo de una placa de circuito impreso para su uso en el sistema de emergencias. Este elemento funcionará como nodo de medida dentro de la red u opcionalmente como Gateway (Fase 6). Debe cumplir una serie de requisitos:
    - Debe disponer de conectividad WPAN y LPWAN
    - Capacidad para carga de batería por distintos métodos, como sistemas cableados como USB (*Universal serial bus*) o sistemas de recolección de energía.
    - Posibilidad de conectar diferentes accesorios sensores para realizar medidas de diferentes parámetros, principalmente, temperatura, humedad, Calidad de aire, gases.
    - Debe ser un sistema modular, los elementos que lo componen deben ser fácilmente sustituibles y fomentar por un sistema de producción sostenible.

El desarrollo de ambas partes se realizará de forma paralela para optimizar el tiempo, se puede observar con mayor detalle en la planificación del trabajo.

### 1.3 Enfoque y método seguido

Si se observa desde el punto de vista del mercado, se puede ver que existe una necesidad urgente en el campo de los sistemas de detección de emergencias. No existe realmente una solución fiable, robusta y generalizada en el mercado actual que cubra estas necesidades.

Respecto a las soluciones propietarias que hay disponibles en el mercado, se puede observar alguna solución que se puede acercar al concepto propuesto en el trabajo, como la que ofrece alguna empresa de seguridad (apartado 2.4), aun así, no dispone de todas las ventajas propuestas en el trabajo y desde el punto de vista de mercado, la barrera de entrada al sector en cuestión es muy frágil por lo que se puede plantear el proyecto propuesto como una posible idea de negocio.

Pero a pesar de sus posibilidades, el trabajo se plantea no como una posible incursión en el mercado. Se ofrece la idea de un sistema global y un servicio básico y público para los usuarios como puede ser el servicio telefónico de emergencias 112 con total independencia de la gestión del proyecto por parte de una empresa privada.

Respecto al enfoque que se ha tomado para el desarrollo del trabajo ha sido el de analizar, combinar y mejorar una serie de productos y tecnologías existentes para la obtención de un producto superior.

Por un lado, como primera estrategia, se plantea el desarrollo de un nuevo producto que cubra las necesidades básicas del sistema a partir de la arquitectura de productos comerciales. Este, es el diseño de un nodo modular capaz de medir diferentes parámetros, con conectividad LPWAN y WPAN, que en posibles mejoras futuras sea capaz de soportar nuevos accesorios como por ejemplo módulos de *energy harvesting*, tales como células fotovoltaicas, o pinzas amperométricas, dotándole la posibilidad de alimentar a los dispositivos de forma indefinida, lo que le daría una gran ventaja a la hora de su instalación y disposición (al ser autónomo podría situarse en lugares difíciles de acceder como por ejemplo el techo de una fábrica).

Por otro lado, de forma paralela, se plantea el análisis, diseño y desarrollo de diferentes arquitecturas de comunicaciones con módulos comerciales, que a pesar de ser muy parecidos al sistema hardware a diseñar, carecen de la flexibilidad y de las propiedades que se quieren implantar en el diseño del trabajo. Este proceso se realizará de forma secuencial a través de diferentes arquitecturas, cada vez más complejas, pero que, en teoría, deberían mejorar la función del sistema de emergencia.

La suma de ambas estrategias aportará una solución completa a todos los niveles, tanto a nivel de concepto, como a nivel de hardware y software de un sistema de emergencias inalámbrico formado a partir de elementos, tanto comerciales como de diseño propio, aportando así una idea global del sistema propuesto.

### 1.4 Planificación del Trabajo

A continuación, se propone la planificación inicial estimada para el desarrollo del trabajo. La planificación se ha dividido en 5 tareas, tantas como entregas a realizar a lo largo del semestre, estas tareas son:

- **Entrega 1:** Resumen e introducción. Definición de los aspectos más globales del trabajo a realizar.
- **Entrega 2**: Estado del arte. Estudio y análisis del panorama actual en el campo de los sistemas de detección de emergencias.
- **Entrega 3:** Diseño e implementación del sistema. Tiene como hitos parciales, el desarrollo hardware. Y por otro lado el desarrollo de la arquitectura del sistema, donde se incluye el firmware de configuración de cada elemento de la arquitectura. Este último bloque se desarrolla en 6 fases complementarias que se realizarán de forma secuencial.
- **Entrega 4:** Memoria. Redacción de la memoria final del trabajo. Se incluyen apartados de conclusiones tras la realización del trabajo, así como posibles trabajos y mejoras futuras sobre el sistema desarrollado, glosario, bibliografía y revisión del trabajo.
- **Entrega 5:** Presentación. Desarrollo de la presentación del trabajo y posible video de demostración del funcionamiento del sistema con un caso práctico.

Las subtareas parciales, con la asignación de recursos y tiempos, así como los hitos parciales, se muestran en las siguientes figuras, estas representan la tabla de tareas y recursos, y el diagrama de Gantt.

| | Nombre | Trabajo | Inicio | Terminado | Predecesor... | Nombres del Recurso |
|---|---|---|---|---|---|---|
| 1 | **Tareas previas** | 20 horas | **10/02/20 8:00** | **13/02/20 10:00** | | |
| 2 | Desarrollo de concepto y propuesta de TFM | 10 horas | 10/02/20 8:00 | 11/02/20 10:00 | | |
| 3 | Busqueda de componentes para desarrollo de TFM | 10 horas | 12/02/20 8:00 | 13/02/20 10:00 | | |
| 4 | **TFM** | 382,5 horas | **20/02/20 19:00** | **20/06/20 20:00** | | |
| 5 | **PEC 1: Resumen e introducción** | 59 horas | **20/02/20 19:00** | **3/03/20 20:00** | | |
| 6 | Desarrollo y aprobación de concepto con tutor | 24 horas | 20/02/20 19:00 | 27/02/20 21:00 | | Eduardo Perez Galan |
| 7 | Resumen | 3 horas | 24/02/20 8:00 | 25/02/20 20:00 | | Eduardo Perez Galan |
| 8 | Índice | 3 horas | 25/02/20 8:00 | 26/02/20 20:00 | | Eduardo Perez Galan |
| 9 | Justificación de trabajo | 8 horas | 26/02/20 8:00 | 29/02/20 11:00 | | Eduardo Perez Galan |
| 10 | Objetivos | 8 horas | 28/02/20 11:00 | 29/02/20 20:00 | | Eduardo Perez Galan |
| 11 | Planificación | 10 horas | 29/02/20 8:00 | 1/03/20 18:00 | | Eduardo Perez Galan |
| 12 | Entrega PEC1 | 3 horas | 2/03/20 9:00 | 3/03/20 20:00 | 6;7;8;9;10 | |
| 13 | **PEC 2: Estado del arte** | 36 horas | **3/03/20 20:00** | **16/03/20 19:00** | **5** | |
| 14 | Documentación y Estudio de sistemas de emergencia actuales | 11 horas | 3/03/20 20:00 | 7/03/20 18:00 | | Eduardo Perez Galan |
| 15 | Documentación y Estudio de comunicaciones inalámbricas iot | 14 horas | 8/03/20 9:00 | 12/03/20 21:00 | | Eduardo Perez Galan |
| 16 | Redacción de entrega | 11 horas | 13/03/20 19:00 | 15/03/20 17:00 | 14;15 | Eduardo Perez Galan |
| 17 | Entrega PEC2 | 0 horas | 16/03/20 10:00 | 16/03/20 19:00 | 16 | Eduardo Perez Galan |
| 18 | **PEC 3: Diseño e implementación** | 170,5 horas | **17/03/20 19:00** | **17/05/20 19:00** | **13** | |
| 19 | **Desarrollo hardware nodo modular** | 49,5 horas | **17/03/20 19:00** | **17/05/20 19:00** | | |
| 20 | Diseño PCB | 20 horas | 17/03/20 19:00 | 29/03/20 18:00 | | Eduardo Perez Galan[50%] |
| 21 | Documentación para fabricacion | 1,5 horas | 29/03/20 18:00 | 30/03/20 20:00 | 20 | Eduardo Perez Galan[50%] |
| 22 | Fabricación | 0 horas | 30/03/20 20:00 | 2/05/20 17:00 | 21 | |
| 23 | Compra de componentes | 2 horas | 4/04/20 7:00 | 4/04/20 18:00 | | Eduardo Perez Galan[50%] |
| 24 | Montaje de prototipo | 6 horas | 2/05/20 17:00 | 5/05/20 20:00 | 22 | Eduardo Perez Galan[50%] |
| 25 | Adaptacion firmware | 8 horas | 5/05/20 20:00 | 10/05/20 17:00 | 24 | Eduardo Perez Galan[50%] |
| 26 | Pruebas en sistema | 8 horas | 10/05/20 17:00 | 16/05/20 17:00 | 25 | Eduardo Perez Galan[50%] |
| 27 | Redacción de resultados | 4 horas | 16/05/20 17:00 | 17/05/20 19:00 | 26 | Eduardo Perez Galan[50%] |
| 28 | **Desarrollo de sistema de emergencias** | 121 horas | **17/03/20 19:00** | **17/05/20 17:00** | | |
| 29 | **Fase 1: Firmware nodos/sensores, parámetros de medida** | 20 horas | **17/03/20 19:00** | **29/03/20 18:00** | | |
| 30 | Desarrollo | 18 horas | 17/03/20 19:00 | 28/03/20 20:00 | | Eduardo Perez Galan[50%] |
| 31 | Documentación de solución y resultados | 2 horas | 29/03/20 9:00 | 29/03/20 18:00 | 30 | Eduardo Perez Galan[50%] |
| 32 | **Fase 2: Comunicación BLE entre nodos y Gateway** | 23 horas | **29/03/20 18:00** | **12/04/20 17:00** | **29** | |
| 33 | Desarrollo | 20 horas | 29/03/20 18:00 | 11/04/20 20:00 | | Eduardo Perez Galan[50%] |
| 34 | Documentación de solución y resultados | 3 horas | 12/04/20 9:00 | 12/04/20 17:00 | 33 | Eduardo Perez Galan |
| 35 | **Fase 3: Comunicación Gateway-Backend Sigfox** | 10 horas | **12/04/20 17:00** | **15/04/20 20:00** | **32** | |
| 36 | Desarrollo | 8 horas | 12/04/20 17:00 | 15/04/20 20:00 | | Eduardo Perez Galan |
| 37 | Documentación de solución y resultados | 2 horas | 12/04/20 17:00 | 12/04/20 19:00 | | Eduardo Perez Galan |
| 38 | **Fase 4: Servidor almacenamiento y BBDD** | 17 horas | **15/04/20 20:00** | **19/04/20 20:00** | **35** | |
| 39 | Desarrollo | 15 horas | 15/04/20 20:00 | 19/04/20 18:00 | | Eduardo Perez Galan |
| 40 | Documentación de solución y resultados | 2 horas | 19/04/20 18:00 | 19/04/20 20:00 | 39 | Eduardo Perez Galan |
| 41 | **Fase 5: Aplicacion de visualizacion de alarmas** | 17 horas | **20/04/20 19:00** | **26/04/20 10:00** | **38** | |
| 42 | Desarrollo | 15 horas | 20/04/20 19:00 | 25/04/20 19:00 | | Eduardo Perez Galan |
| 43 | Documentación de solución y resultados | 2 horas | 25/04/20 19:00 | 26/04/20 10:00 | 42 | Eduardo Perez Galan |
| 44 | **Fase 6: Optimizacion de arquitectura** | 34 horas | **26/04/20 10:00** | **17/05/20 17:00** | **41** | |
| 45 | **6.1 Cambio de topología BLE estrella a mesh** | 17 horas | **26/04/20 10:00** | **7/05/20 20:00** | **41** | |
| 46 | Desarrollo | 15 horas | 26/04/20 10:00 | 5/05/20 20:00 | | Eduardo Perez Galan[50%] |
| 47 | Documentación de solución y resultados | 2 horas | 5/05/20 20:00 | 7/05/20 20:00 | 46 | Eduardo Perez Galan[50%] |
| 48 | **6.2 Incorporación capa de comunicación LPWAN** | 17 horas | **7/05/20 20:00** | **17/05/20 17:00** | **45** | |
| 49 | Desarrollo | 15 horas | 7/05/20 20:00 | 16/05/20 19:00 | | Eduardo Perez Galan[50%] |
| 50 | Documentación de solución y resultados | 2 horas | 16/05/20 19:00 | 17/05/20 17:00 | 49 | Eduardo Perez Galan[50%] |
| 51 | Entrega PEC3 | 0 horas | 17/05/20 19:00 | 17/05/20 19:00 | 19;28 | |
| 52 | **PEC 4: Memoria** | 87 horas | **18/05/20 19:00** | **9/06/20 21:00** | **18** | |
| 53 | Redacción | 50 horas | 18/05/20 19:00 | 3/06/20 21:00 | | Eduardo Perez Galan |
| 54 | Posbiles Mejoras | 5 horas | 18/05/20 19:00 | 20/05/20 20:00 | | Eduardo Perez Galan |
| 55 | Conclusiones | 6 horas | 4/06/20 19:00 | 6/06/20 11:00 | 53 | Eduardo Perez Galan |
| 56 | Glosario | 6 horas | 4/06/20 19:00 | 6/06/20 11:00 | 53 | Eduardo Perez Galan |
| 57 | Revision | 20 horas | 4/06/20 19:00 | 9/06/20 21:00 | 53 | Eduardo Perez Galan |
| 58 | Entrega PEC4 | 0 horas | 9/06/20 21:00 | 9/06/20 21:00 | 53;55;56;57 | |
| 59 | **PEC 5: Presentacion** | 30 horas | **12/06/20 19:00** | **20/06/20 20:00** | **52** | |
| 60 | Presentación | 15 horas | 12/06/20 19:00 | 15/06/20 20:00 | | Eduardo Perez Galan |
| 61 | Video demo | 15 horas | 15/06/20 20:00 | 20/06/20 20:00 | 60 | Eduardo Perez Galan |
| 62 | Entrega PEC5 | 0 horas | 20/06/20 20:00 | 20/06/20 20:00 | 61 | |

*Tabla 1: planificación de tareas y recursos*

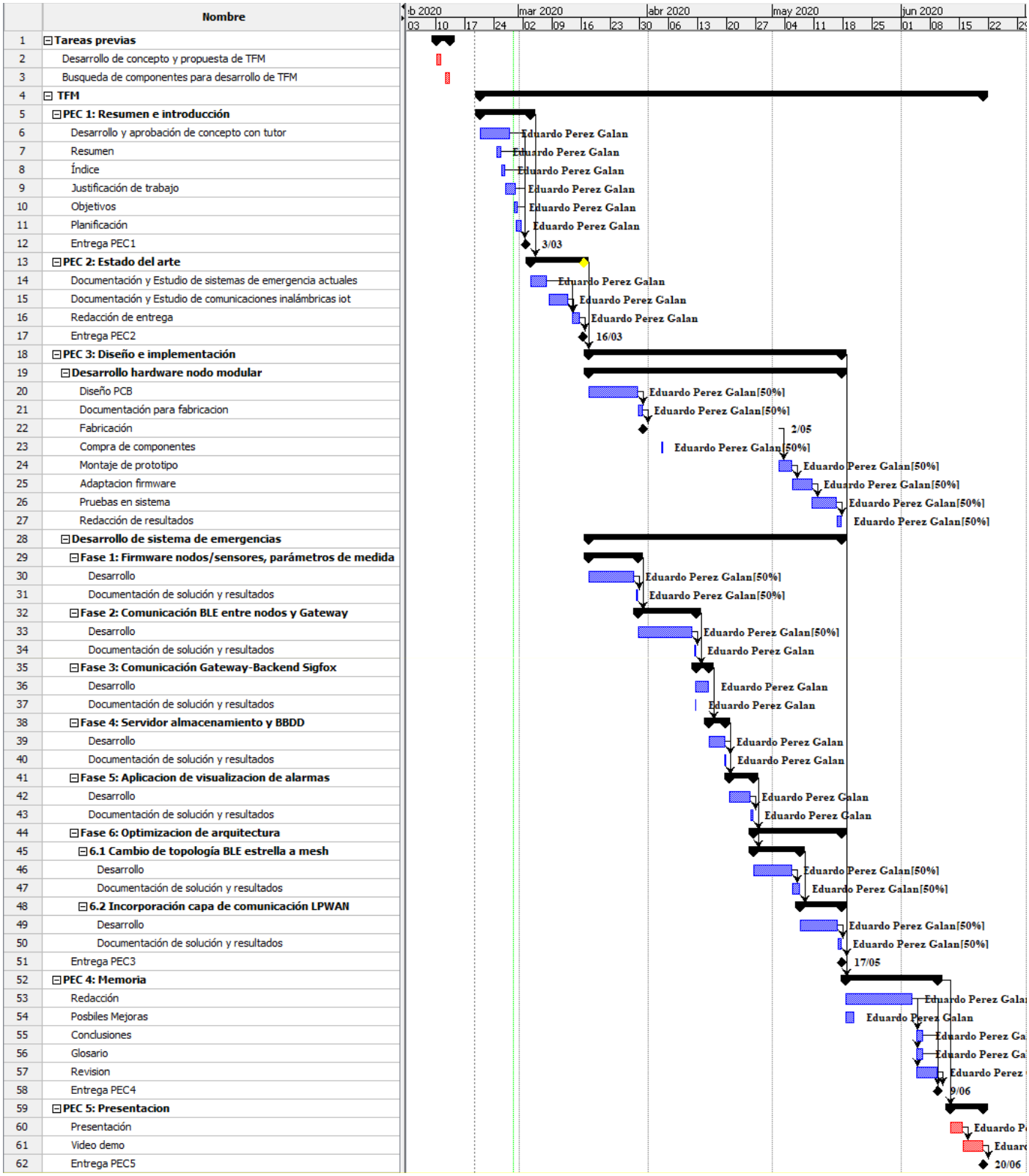



*Tabla 2: Diagrama de Gantt*

Respecto a la gestión de tiempos, se ha modificado el calendario laboral en el programa de proyectos para adaptarlo a la disponibilidad de los recursos temporales. En el caso en cuestión, se ha propuesto una estimación de lunes a viernes de 2 horas, mientras que sábados y domingos se ha propuesto una jornada de 6 horas, con un total de 22 horas por semana.

### 1.5 Breve sumario de productos obtenidos

Como productos finales obtenidos a lo largo del desarrollo del presente proyecto, se pueden clasificar en dos bloques:

A nivel de hardware se han obtenido principalmente dos productos:

- Un diseño hardware de un sistema electrónico modular con funciones de nodo y gateway LoRa dentro de una vivienda. Dicho modulo consta de una placa principal donde se encuentra la unidad de procesamiento, y varios módulos accesorios intercambiables con las funciones de: Alarma sonora, sensor medioambiental y transceptor ISM (*Industrial, Scientific and Medical frequency band*) 868 MHz para comunicaciones LoRa.

- Un sistema de emergencias funcional compuesto por diferentes modelos de placas comerciales adaptadas para el desarrollo de las funciones propuestas en el proyecto.

A nivel de firmware/software se han desarrollado los siguientes elementos necesarios para el funcionamiento del sistema:

- Desarrollo de programación en Arduino para nodo/sensor y gateway/LoRa, para la medida de parámetros críticos como temperatura y niveles de gases, así como los medios de comunicación para transmitir la alarma a través del sistema por tecnologías WPAN y LPWAN.

- Desarrollo de código en Python para la programación de gateway Sigfox. Encargado de enlazar toda la información procedente de los *gateway* LoRa y transmitirla a la nube a través de Sigfox.

- Desarrollo de un servidor AWS (*Amazon Web Services*) con BBDD (Bases de datos) en mongodb para la visualización de alarmas a través de cliente Web por parte del usuario final.

Todos estos elementos permiten la implementación de un sistema inalámbrico de emergencias con capacidad de detección de incendios gracias a la medida de parámetros del entorno como temperatura o concentración de gases. También permitirá al usuario dar el aviso de otro tipo como alarmas como, por ejemplo, emergencias médicas, pulsando uno de los botones disponibles en cada nodo. El sistema aparte de avisar directamente al centro de emergencias dispondrá de un zumbador o Buzzer para la emisión de alarma sonora ante la activación de una alarma, pudiendo alertar a los habitantes de la vivienda.

## 1.6 Breve descripción de los otros capítulos de la memoria.

La memoria se divide en los siguientes capítulos:

- **Estado del arte**: Resumen y panorama actual de los sistemas de emergencias, así como de las tecnologías relacionadas con las redes de sensores inalámbricas. Después se realiza un estudio con ejemplos de sistemas similares al propuesto como trabajo, para finalizar con una comparativa de todos los ejemplos y las contribuciones realizadas por la nueva arquitectura presentada.

- **Fundamentos teóricos de redes de sensores**: Introducción a los sistemas de comunicación basados en redes de sensores, elementos que los forman y principales características que deben incluir.

- **Desarrollo de PCB** (*Printed Circuit Board*) **modular con conectividad BLE** (*Bluetooth Low Energy*)**, WiFi** (*Wireless Fidelity*) **y LoRa**: Capitulo que recoge todos los pasos seguidos para el desarrollo hardware de un sistema electrónico modular con funciones de nodo y gateway.

- **Desarrollo de sistema de emergencias**: Capitulo que introduce al concepto de la arquitectura propuesta y cada uno de los niveles que lo componen. Después describe el desarrollo de toda la programación y configuraciones necesarias sobre cada uno de los elementos participes del sistema.

- **Pruebas del sistema completo y análisis de resultados**: En este apartado se realizarán pruebas sobre el sistema desarrollado, permitiendo conocer las fortalezas y debilidades de este en un entorno real.

- **Valoración económica del trabajo:** En este capítulo se desglosa el coste total económico del producto obtenido en el trabajo.

- **Conclusiones**: Capitulo que reunirá un repaso sobre los objetivos cumplidos, resultados obtenidos, puntos inicialmente propuestos que no se han podido cumplir y posibles debilidades del sistema. A partir de las conclusiones obtenidas, se proponen posibles trabajos e ideas de mejora del sistema.

- **Glosario**: Acrónimos del documento

- **Bibliografía**: Listado de todos los documentos consultados durante el desarrollo del trabajo

- **Anexos**: incluye documentación relacionada con el producto desarrollado: esquemas, planos de montaje, listado de componentes utilizados y código para la configuración de cada uno de los dispositivos de la arquitectura.

# 2. Estado del arte

En el siguiente capítulo se presentará el concepto de emergencia y se analizarán los sistemas de detección de incendios y tecnologías de detección disponibles en la actualidad. El siguiente apartado será un pequeño resumen de las principales tecnologías de comunicaciones inalámbricas WPAN y LPWAN. Finalmente se propondrán unos ejemplos de últimas tecnologías en el sector y se analizará las mejoras introducidas en el presente trabajo respecto a las mismas.

## 2.1 Generalidades

### 2.1.1 Concepto de emergencia

Se puede definir emergencia como una situación o acontecimiento no deseado que pudiera poner en peligro la integridad de los efectos materiales, habitáculos o incluso las personas que los albergan, exigiendo una evacuación y una acción de respuesta inmediata **[6]**.
Este tipo de situaciones pueden presentarse en muchos ámbitos, empresa, domicilio, entornos naturales, etc. Y pueden presentarse debido a diferentes procedencias o causantes:

- Emergencias ocasionadas por intervenciones antropogénicas:
  - Emisiones
  - Fugas
  - Incendios
  - Explosiones
  - Amenazas de bomba
- Emergencias Sanitarias
- Emergencias Sociales
- Emergencias radiactivas

### 2.1.2 Sistemas detección de incendios

Se define como detección de incendios al hecho de descubrir y notificar de la existencia de un incendio en un determinado lugar, para así poder tomar acciones de respuesta y salvaguardar la vida de las personas y los bienes materiales. Los sistemas de detección deben ser sistemas que localicen de forma precisa el origen del incendio y comunicarlo de forma fiable.

La detección de un incendio puede realizarse por: **[10]**

- Detección humana
- Sistema de detección automática
- Sistemas mixtos.

**Detección humana:** La detección del incendio es responsabilidad exclusiva de las personas. El proceso de detección se puede clasificar en dos periodos de tiempo. Durante el día, la simple presencia de las personas en las distintas áreas de la instalación permite una detección de las áreas y zonas visibles. Por la noche será responsabilidad del personal de vigilancia, el cual, realiza supervisiones periódicas de las instalaciones. Este sistema de detección se

caracteriza por un tiempo de respuesta muy alto y con una dependencia total de la periodicidad de las supervisiones realizadas por la vigilancia.

**Detección automática de incendios:** Estos sistemas permiten la detección y localización del incendio de una forma automática. De existir contramedidas de respuesta, estas pueden activarse también de forma automática según la secuencia del plan de emergencia programado en la central.
Estos sistemas se caracterizan por tener un tiempo de detección y, por lo tanto, un tiempo de respuesta mucho menor que en el caso de la detección humana, aunque en este caso también tienen una mayor probabilidad de falsas alarmas.

Los principales componentes que intervienen en un sistema automático de detección de incendios son:

- Central de señalización y mando
- Pulsadores manuales
- Líneas de comunicación
- Sensores o detectores de incendios automáticos
- Actuadores
- Elementos auxiliares

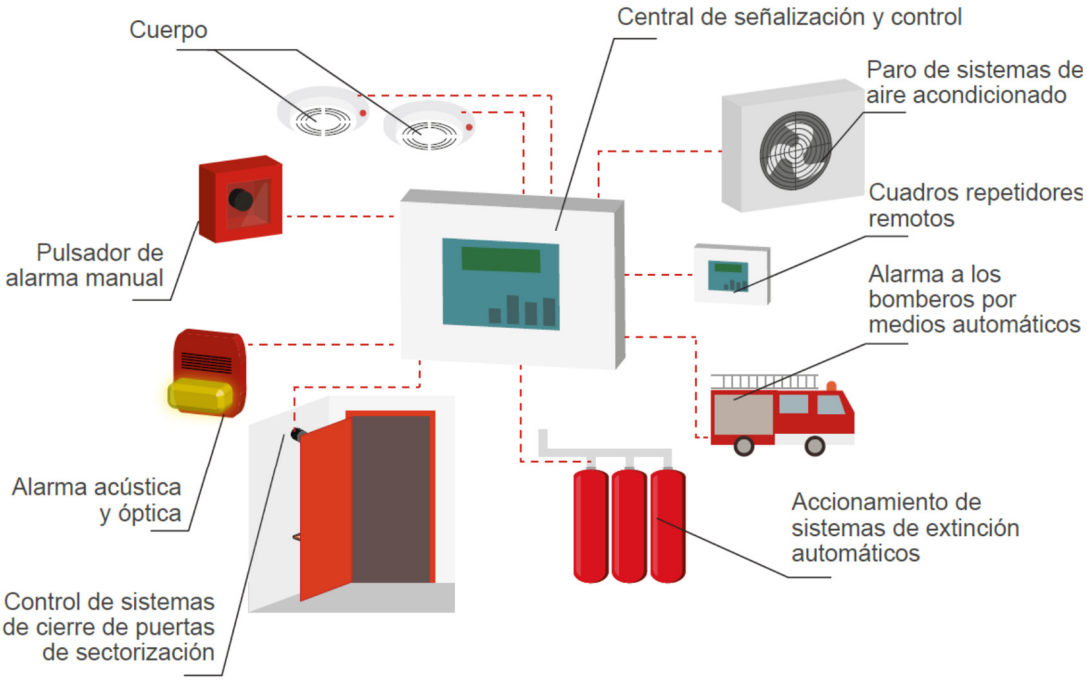



*Figura 3: Componentes de un sistema de detección automática de incendios [7]*
*Se puede observar que todos los componentes mostrados están formados por elementos sensores, procesamiento, comunicaciones y de actuación ante la detección positiva de un incendio*

## 2.2 Métodos de detección de incendios. Sensores.

Al producirse un incendio, existen varios elementos que se generan a partir del proceso químico, como son gases, humo, aumento de radiación y un incremento de la temperatura. Los principales elementos en un sistema de incendios automático son los detectores de estos productos, también se les puede denominar elementos sensores.

Tal y como se ha visto, existen múltiples métodos para la detección de un incendio, la elección del método más apropiado viene dado por una serie de condiciones:

- Número de personas o valor de materiales que contiene el recinto
- Posibilidad de vigilancia constante por personas.
- La velocidad de respuesta requerida
- La Fiabilidad requerida para el sistema
- Coherencia con el resto del sistema de emergencias global.
- Coste económico del proyecto

En la siguiente figura, se puede observar en las fases en las que actúa cada uno de los tipos de detectores y que elemento analiza para comprobar el riesgo de incendio.

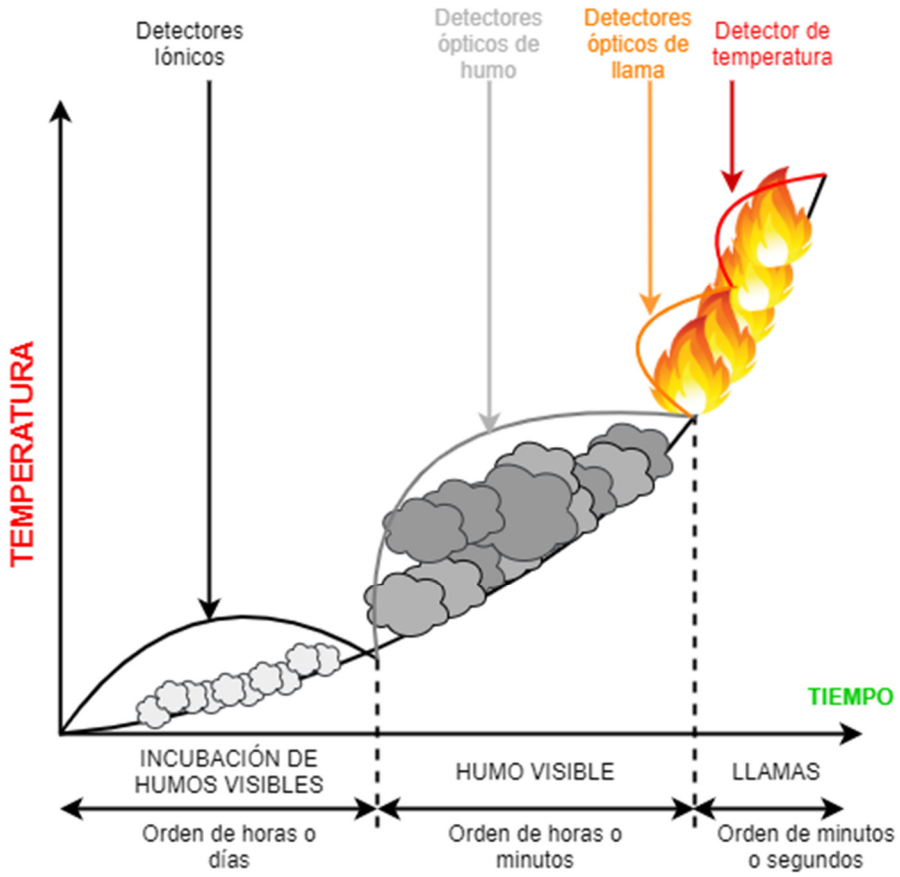



*Figura 4: Fases de actuación de los detectores.[8]*
*Existen sensores de varios tipos, cada uno de ellos puede actuar en una fase distinta en el proceso de creación de un incendio, pudiendo detectar distintas fases del humo generado o un incremento de temperatura.*

### 2.2.1 Detectores iónicos de humos [11]

Se trata de dispositivos que detectan gases producidos en la combustión, estos pueden ser visibles o invisibles.

Se denominan iónicos o de ionización porque su funcionamiento se fundamenta en el uso de una cámara formada por dos placas ionizadas por un elemento radiactivo (Americio 241), esto provoca una ionización del aire que circula en el espacio existente entra las placas y generando una corriente eléctrica de forma continua en estado de reposo del detector. Cuando se produce el incendio, las partículas que se liberan producen una interferencia en la ionización en la cámara del detector, produciendo una reducción de la corriente producida entre las placas, que cuando sea medida por el circuito eléctrico, se dará la condición de alarma.

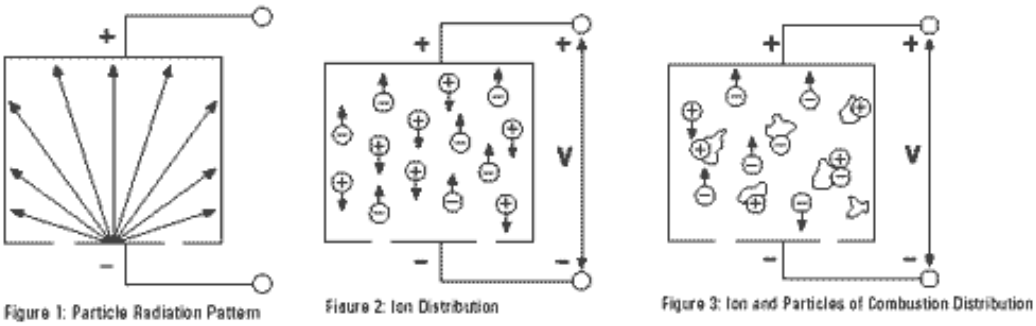



*Figura 5: Funcionamiento de un detector iónico de humos [9]*

La sensibilidad de este tipo de detectores es muy variable, existen muchos factores que afectan en la detección de la alarma y aún más en la primera fase de un incendio. Unos posibles ejemplos que pueden provocar una falsa alarma son: humos de tubo de escape, calderas o cualquier tipo de humo que no proceda de la fuente de la combustión.

### 2.2.2 Detectores de humos visible o fotoeléctricos

Estos detectores se basan en el oscurecimiento que se genera al producirse una cantidad de humo, también se produce por la difusión de luz por el efecto Tyndall, que se basa en el principio de dispersión de la luz.

En el interior del detector, se encuentra un LED emisor de luz y un fotosensor. La luz que se emite incide en un area donde es captada por el fotosensor. Esta es la condición de funcionamiento sin alarma. La condición de alarma se genera cuando el humo producido en el incendio entra en la cámara del detector y produce un oscurecimiento del medio de propagación, al producirse una dispersión de la luz hace que esta, se refleje hacia el fotosensor.

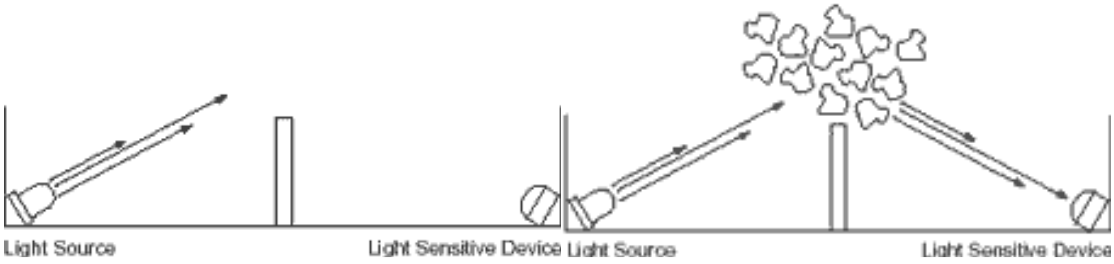



*Figura 6: Funcionamiento de un detector fotoeléctrico [9]*

El principal inconveniente que presentan este tipo de detectores es que se trata de dispositivos con un diseño muy complejo ya que requieren de una fuente de emisora de luz, una célula captadora y un equipo eléctrico muy complejo. También se producen falsas alarmas principalmente debidas a las partículas de polvo.

### 2.2.3 Detectores térmicos [12].

Se pueden encontrar principalmente de dos tipos según la detección de la temperatura:

- Por temperatura fija
- Por gradiente de temperatura.
- Combinados

Los detectores de temperatura fija son los dispositivos de detección más antiguos. Se trata de dispositivos que actúan solamente cuando se alcanza una temperatura umbral. Su funcionamiento se basa en la deformación de un bimetal (Al aumentar la temperatura el material se deforma hacia un contacto fijo, cerrando un circuito eléctrico y activando la alarma). o en la fusión de una aleación (Formados principalmente por dos conductores metálicos tensados se parados por un termofusible a cierta temperatura se funde el aislamiento y los dos conductores entran en contacto).
Los detectores de gradiente de temperatura se denominan termovelocimétricos. Este tipo de dispositivos son más habituales en las instalaciones, son ideales para instalar en áreas donde se prevé un incremento rápido de temperatura en caso de incendio.

### 2.2.4 Detectores de radiación.

Se trata de dispositivos que detectan las radiaciones infrarrojas y ultravioletas que producen las llamas. Están compuestos por filtros ópticos, células captadoras y componentes electrónicos para amplificar las señales.
Se instalan principalmente en espacios abiertos o recintos con techos elevados, además deben instalarse con cierta separación de fuentes de radiación para evitar falsas alarmas.

## 2.3 Comunicaciones inalámbricas orientadas a comunicación de sensores.

En la actualidad, debido principalmente al crecimiento de la tecnología orientada a IoT, han surgido varias tecnologías y estándares de comunicación inalámbricos orientados a redes de sensores. Estas tecnologías se pueden caracterizar y clasificar con tres aspectos: Consumo, cobertura, y tasa de transferencia de datos. Pero en común deben tener también una serie de características como: bajo coste, escalabilidad, flexibilidad, HW sencillo y tolerancia a fallos. En el presente trabajo se utilizarán dos tecnologías muy diferentes, que, combinándolas, se pretende diseñar una red de sensores hibrida que aprovechará al máximo las bondades de cada una.

### 2.3.1 Comunicaciones WPAN

Wireless Personal Area Network o WPAN se trata de un estándar orientado a comunicaciones inalámbricas entre distintos tipos de dispositivos a corta distancia (unos 10 metros con obstáculos) del gateway, que es el dispositivo que establece una conexión con el mundo exterior. Están principalmente pensadas para comunicaciones personales.

Las principales tecnologías que pertenecen a este estándar son:

- **Bluetooth:** basada en el estándar IEEE (*Institute of Electrical and Electronics Engineers*) 802.15.1 [13], es una tecnología que ha ido evolucionando continuamente hasta conseguir unas tasas de transmisión y de consumo energético óptimos, haciéndola la tecnología WPAN más extendida. A partir de la extensión de Bluetooth 4.2 [14] está disponible el uso de un modo de funcionamiento denominado Bluetooth Low Energy el cual está adaptado para las nuevas exigencias orientadas a la comunicación IoT, esto le permite reducir aún más su consumo y por lo tanto aumentar la eficiencia energética del sistema. Actualmente la tecnología se encuentra en la extensión 5.0[15].
  Características:
  - Banda de Frecuencia: 2.4 GHz
  - Rango: 100-200 metros en espacios abiertos 10 a 20 metros en interior con obstáculos.
  - Velocidad: Hasta 1Mbits/s (Bluetooth 4.2), 2Mbits/s (Bluetooth 5.0)
  - Topología de red: punto a punto, estrella, malla.

- **Zigbee:** Tecnología basada 802.15.4 [16]. es definida como una LR-WPAN o *Low Rate Wireless Personal Area Networks.* Está orientada a sistemas de comunicación seguros con una baja tasa de transferencia de datos y bajo consumo.
  Características:
  - Banda de Frecuencia: 868 MHz, 915MHz y 2,4GHz
  - Acceso al medio mediante CSMA (*Carrier Sense Multiple Access*). Se comprueba que alguien este transmitiendo, si el canal esta libre, se inicia la transmisión.
  - Topología de red: Estrella, Malla, Árbol.
  - Rango: 150 metros
  - Velocidad máxima de transmisión: 250 kbps
  - Protocolo propietario, por lo tanto, baja compatibilidad entre dispositivos.
  - Se trata de una red con diferentes tipos de dispositivos:
    - Final: Son los dispositivos sensores, no realizan ninguna otra función y no pueden enrutar información de otros nodos. Para transmitir información deben estar conectados a un dispositivo enrutador o un coordinador.
    - Enrutador: Funciona como puente entre la información procedente de los dispositivos sensores conectados a él con el sistema coordinador.

        - Coordinador: Es el dispositivo que funciona como puerta de la red Zigbee hacia otras redes externas. Hay un coordinador por red.

- **Infrarrojos, IR (*infrared*) [17]:** Tecnología que permite a dos dispositivos establecer una comunicación punto a punto mediante el uso de LED's que emiten haces de luz con una longitud de onda infrarroja. Este tipo de sistema de comunicación esta principalmente pensado para la comunicación entre electrodomésticos, telefonía, ordenadores o equipamiento médico.
  Características:
    - Rango: 10 metros
    - Topología: punto a punto
    - Velocidad máxima: 10Mbits/s

### 2.3.2 Comunicaciones LPWAN

LPWAN o Low Power Wide Area Network son un conjunto de tecnologías de redes inalámbricas que permiten la transmisión de pequeñas cantidades de datos a través de largas distancias a un coste energético muy bajo. Para ello, dichas transmisiones en muchas ocasiones quedan acotadas en pequeños periodos de tiempo, con posteriores ventanas de recepción.
Existen diferentes opciones:

- Espectro Licenciado:
    - **NB-IoT [18]**: (*Narrowband Internet of things*), es una tecnología LPWAN que usa bandas ISM que están licenciadas y puede funcionar junto con las redes de telefonía móvil actuales como 4G u otras tecnologías que se encuentran casi obsoletas y próximas a desaparecer como por ejemplo las redes 2G. Por lo tanto, al utilizar una tecnología consolidada y conocida, se produce un ahorro en el coste de implantación y certificación de las operadoras.
      Características:
        - Banda de frecuencia: Celular- Espectro licenciado < 900MHz
        - Rango: 15 km.
        - Velocidad de transferencia: 150 kbps

- Espectro no licenciado:
    - **LoRa [19]:** es una tecnología que usa bandas ISM no licenciadas. Utiliza una modulación patentada por Semtech denominada CSS (*Chirp spread spectrum*) que le permite una mayor robustez ante interferencias o ruido. El transmisor modifica la frecuencia a la que emite y eso permite que los receptores puedan realizar decodificaciones de las señales recibidas a pesar de su atenuación, por lo tanto, a cada frecuencia le corresponde una distancia y velocidad diferentes de transmisión.
      Características:
        - Comunicación: Bidireccional
        - Banda de frecuencia: 433 y 868 MHz (Europa), 915MHz (América).

- Velocidad de transferencia: 37,5 Kbps
- Canales: 10(Europa)
- Cobertura máxima: 5 km (Urbano)-15km (rural)
- Topología de red:
  - Malla
  - Estrella de estrellas. La comunicación en esta topología permite conocer el espacio temporal de recepción de cada estación y así poder realizar una localización geográfica del nodo transmisor.
- Tipos de dispositivos:
  - Clase A: Orientados a transmisión exclusivamente.
  - Clase B: Orientado a transmisiones en ventanas temporales
  - Clase C: Dispositivos orientados a la escucha continua para recepción de información. Sin ninguna limitación es su consumo.

- **Sigfox [20]:** Se trata de una tecnología LPWAN propietaria que dispone de una infraestructura ya consolidada por toda Europa y en algunas importantes áreas a nivel Global. Esta infraestructura consta una red de estaciones base y de servidores dedicados a la gestión de los datos recibidos. La comunicación es bidireccional, aunque asimétrica, ya que la capacidad de recepción de los dispositivos es menor que la de transmisión.
  Características:
  - Capa física UNB (*Ultra Narrow Band*)
  - Transmisión: 100 bps. 140 mensajes de 12 bytes.
  - Recepción: 600 bps. 4 mensajes de 8 bytes.
  - Banda de frecuencia: ISM 868 MHz
  - Canales: 400
  - Rango: 5km (Urbano) - 50km (Rural)

### 2.4 Ejemplos de aplicación para redes de sensores mixtos

A continuación, se expondrán publicaciones sobre sistemas de sensores relacionados con el presente trabajo con el fin de analizar y comparar las mejoras introducidas por el sistema propuesto.

#### 2.4.1 Ejemplo 1: Low Energy Smart Cities Network with LoRa and Bluetooth

El primer documento [21], se publicó en el año 2019 en la conferencia internacional MobileCloud. Se trata de un análisis de una arquitectura hibrida formada por dos tecnologías diferentes, WPAN y LPWAN.
La tecnología WPAN seleccionada es Bluetooth Low Energy, con ella se consigue una gran conectividad entre dispositivos cercanos a un consumo energético bajo que le permite una autonomía de años con una sola carga de batería. También dispone de gran variedad de topologías de comunicación: P2P, Estrella y Mesh.
Por otro lado, la tecnología LPWAN seleccionada es LoRaWAN. Con ella se consiguen comunicaciones con una gran cobertura, a nivel nacional o incluso global pudiendo implementar cualquier protocolo. Las bandas de frecuencias en

la que se comunica están en el rango no licenciado ISM y permite comunicaciones bidireccionales, gran movilidad e incluso Geolocalización aproximada del dispositivo transmisor.
Respecto a la estructura de la arquitectura analizada en el documento, se menciona una topología en estrella para conectar los dispositivos BLE con el gateway programado con un algoritmo denominado LoraEdge, este dispositivo dispondrá de conectividad LoRa para poder transmitir y recibir datos procedentes de otro dispositivo con conectividad LoRa haciendo de estación base. Este último dispondrá del algoritmo LoraFog que permitirá la comunicación de las capas inferiores con la nube.
Con la combinación de ambos algoritmos, se consigue la correcta integración y comunicación del sistema.

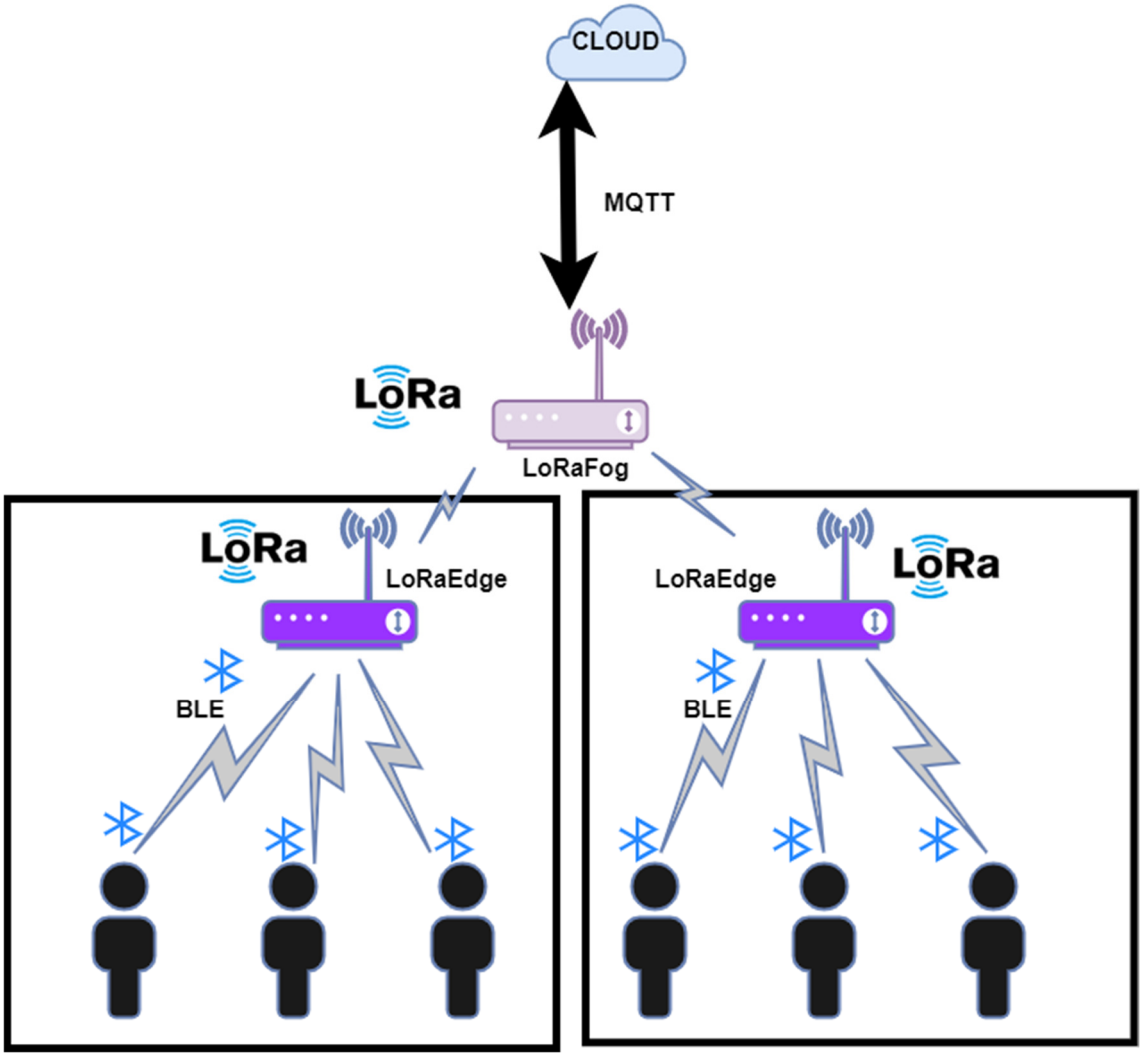



*Figura 7: Arquitectura LoRa + BLE [21]*
*La arquitectura consta de 3 niveles de comunicación, así la primera capa usa una comunicación WPAN como BLE, para luego ser enrutada y transmitida a la nube a través de varios niveles de LoRa*

Con la arquitectura correctamente configurada, se procedió a realizar una prueba real para validar y verificar el comportamiento de la red de comunicación. Para ello se utilizaron dispositivos Lopy4 haciendo las funciones de los dos niveles de

gateway y como dispositivo motas se usaron Pysense, los cuales disponen de diferentes sensores para analizar el entorno.
Los resultados fueron:

- **Cobertura:** Fue posible el envío de mensajes hasta a 3 km de distancia con zonas de visibilidad directa, 1,5 km zonas sin visibilidad y 1km en zonas cerradas.

- **Escalabilidad**: El número de motas que pueden conectarse a un mismo Gateway para transmitir una vez al día y con una tasa de éxito en mensajes de 95% es de 8000 o más. Si se aumenta el número de mensajes, por ejemplo, a 1 mensaje cada hora al mismo Gateway el rendimiento baja y se produce un mayor número de colisiones, por lo que el número máximo de motas conectadas se reduce a 4000.

### 2.4.2 Ejemplo 2: Fire Detection. Semtech

En el siguiente ejemplo la empresa Semtech [22], encargada en la creación de la tecnología LoRaWAN. Propone una posible aplicación para dicha tecnología en lo referente a la detección de incendios en edificios.

1. Se propone un sistema de comunicación inalámbrico exclusivamente compuesto por dispositivos con conectividad LoRa. Todas las motas están conectadas en estrella a un Gateway por LoRa.
2. Cada cierto periodo de tiempo transmite información al Gateway que tiene acceso al servidor en la nube.
3. Se envía información al servidor de la red donde una aplicación centralizada analiza los datos recibidos.
4. Por último, la aplicación del servidor envía los mensajes de alarma a los propietarios del edificio o al personal de emergencias vía móvil u ordenador

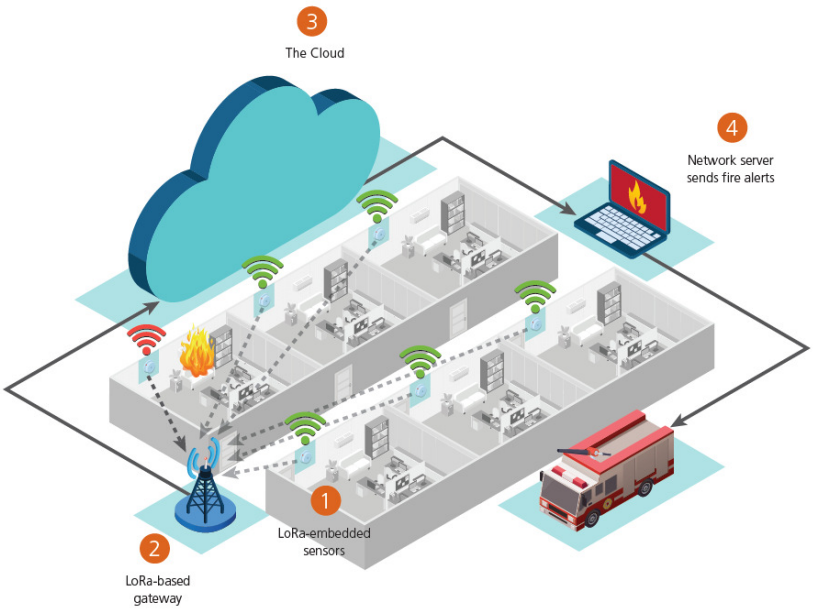



*Figura 8: Propuesta sistema de detección de incendios a través de LoRa [22]*

### 2.4.3 Ejemplo 3: Solución comercial Siterwell

A continuación, como ejemplo se propone la solución aportada por la empresa Siterwell [23], una empresa china especializada en el desarrollo de soluciones de seguridad global para edificios, entre los que se incluye sistemas antirrobo, Smart home o sistemas de detección de incendios. Dentro de este campo, existen varios sistemas orientados a diferentes áreas, como son sistemas de detección convencionales o sistemas inteligentes basados en sistemas de sensores inalámbricos por conectividad LPWAN.
Disponen una gran oferta de dispositivos, tanto por parámetro de detección como por el tipo de conectividad. Los principales sistemas LPWAN utilizados son LoRa, NB-IoT y SigFox.

A continuación, se muestra un diagrama de la arquitectura

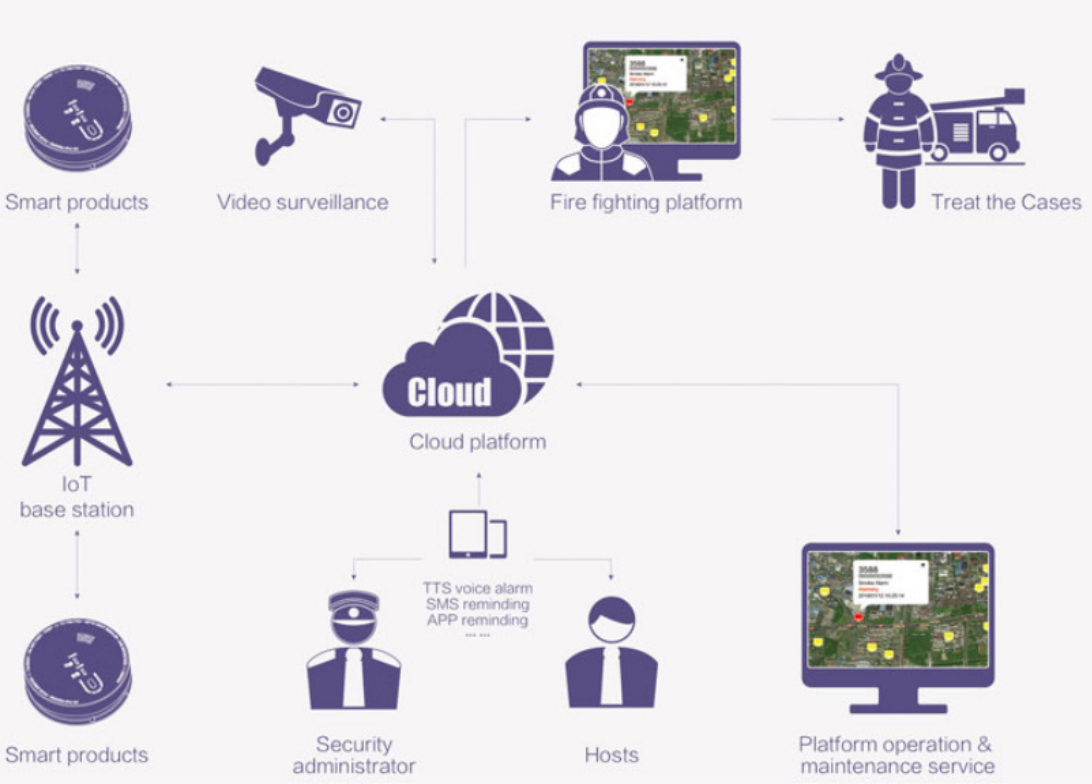



*Figura 9: Arquitectura de sistema detección de incendios por Siterwell* [23]

Como se puede observar en la figura 9, la arquitectura propuesta es muy similar a la propuesta de Semtech, con la diferencia de que dispone de varias opciones en la tecnología de comunicación. A través de software, se pueden realizar diferentes configuraciones para que el aviso vaya directamente al dueño o personal de seguridad, como también al centro de emergencias.

### 2.4.4 Ejemplo 4: Mobile Fire alarm using WSN (*Wireless Sensor Network*).

El siguiente ejemplo, se trata de una publicación [24] presentada en el "*IEEE international on computational inteligence and informatics"* del año 2016. En el mismo, se plantea un sistema de detección con conectividad exclusiva WLAN. En particular, se usaría Wireless LAN (*Local Area Network*) 802.11 b/g. Una tecnología ya convencional que existe en cualquier vivienda.

El planteamiento propuesto se centra en la implementación de unos dispositivos sensores con conectividad inalámbrica WiFi como forma de comunicación de alarmas al usuario a través del móvil.
En el documento se especifica que el dispositivo sensor deberá estar formado por:

1. Un bloque formado por un módulo sensor: Humedad, temperatura y detección de gases.
2. Unidad de almacenamiento con unidad de procesamiento basada en microcontrolador
3. Unidad de transmisión/recepción
4. Unidad de alimentación.

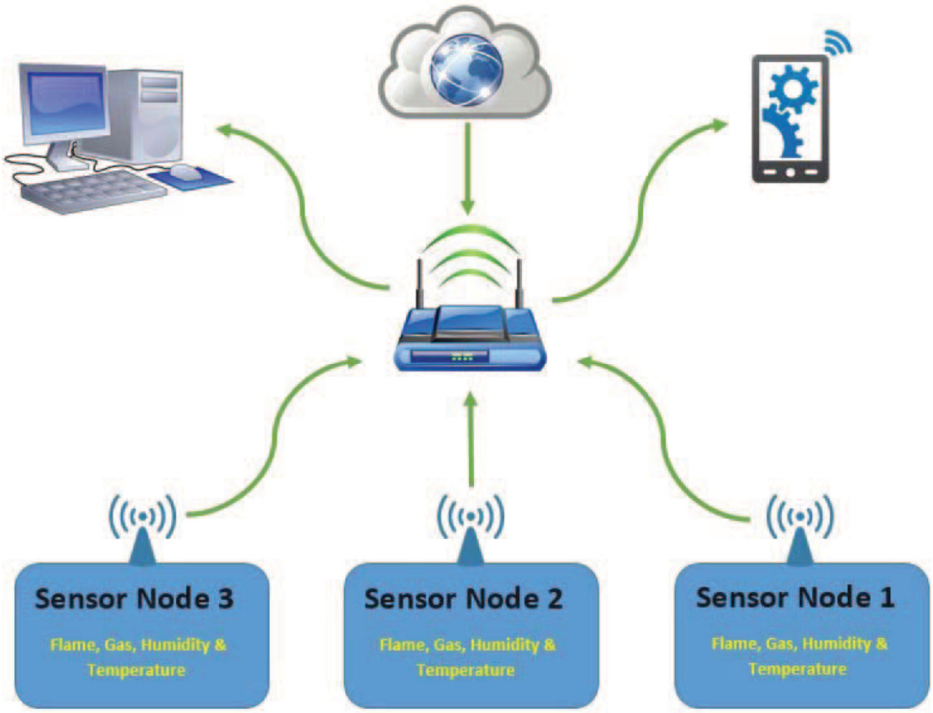



*Figura 10: Arquitectura de detección basada en conectividad WiFi* [24]

La arquitectura de la red, en este caso, se basa principalmente en una red doméstica en la que cada uno de los dispositivos sensores están repartidos por la vivienda y conectados a través de un Gateway, el router de la vivienda, que proveerá de conectividad exterior a internet, o conectividad a cualquier dispositivo cliente de la vivienda como por ejemplo un ordenador o un móvil para poder recibir el aviso.

### 2.4.5 Ejemplo 5: Alarma de seguridad comercial para uso domestico

Como último ejemplo, se propone un sistema de seguridad domestico comercial. Existen numerosas empresas en España que ofrecen un servicio integral de seguridad y uno de los servicios que se incluye opcionalmente son módulos de detección de incendios a partir de sensores de gas o temperatura.
Como se muestra en la figura 11, la arquitectura de este sistema **[25]** se basa en un sistema de comunicaciones hibrido, formando una red local interna entre los diferentes dispositivos conectados con la central a partir de una tecnología propietaria basada en WPAN como puede ser Zigbee, Bluetooth o WLAN

(*Wireless Local Area Network*) como WiFi, de tal manera que se puede transmitir toda la información captada desde los diferentes puntos de control de la casa. La central de control transmitirá toda la información captada por los sensores a la central de vigilancia, que será la que juzgue la necesidad de comunicar con las autoridades pertinentes de la emergencia, por lo tanto, en este caso estamos hablando de una detección mixta, que combina tanta detección humana como detección automática. El método de transmisión de información entre la central de la vivienda con la de emergencias será a través de tecnología móvil, GSM, GPRS o LTE dando la posibilidad de enviar grandes cantidades de información entre ambas centrales, como pueden ser imágenes o ficheros de audio.

Existen otros servicios en lo que a conectividad se refiere ofrecidos por algunas compañías, como es por ejemplo el servicio que ofrece Securitas Direct **[26]**, el cual, detecta sistemas de inhibición de frecuencia y acto seguido, transmite la alarma a través de una red alternativa de alta seguridad basada en un protocolo propietario denominado ATM, que es muy similar a la tecnología Sigfox.

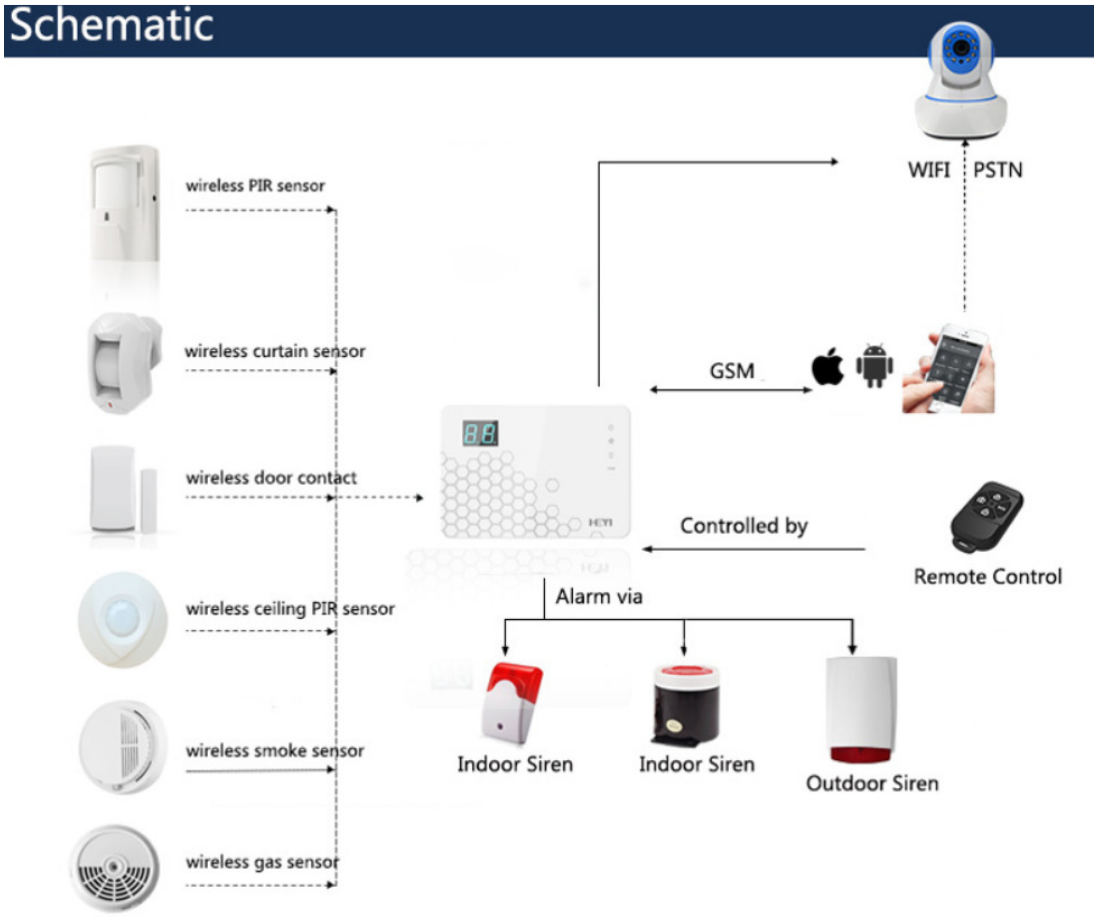



*Figura 11: Diagrama de sistema de seguridad doméstica basada en GSM/GPRS/LTE* [25]
*Todos los accesorios de medida disponen de conectividad inalámbrica para la comunicación con la estación central que actúa como puerta de enlace a la red móvil y al servidor de cliente*

## 2.5 Contribución del trabajo al area de estudio.

Como se ha podido observar en los anteriores ejemplos, las redes de sensores basadas en tecnologías como la gran mayoría de los sistemas propuestos están todavía en fase de investigación, aun así, dos de los ejemplos propuestos son sistemas comerciales que ya están totalmente implantados en el hogar actual. Cabe destacar varios ejemplos. En el ejemplo 1 se ha podido ver como la unión de dos tecnologías tan diferentes como bluetooth y LoRa permiten crear una red de sensores con las características más favorables de ambas. Proporcionando un sistema, flexible y con una gran cobertura a un precio menor.

Por otro lado, en el ejemplo 2, se puede observar que se están desarrollando sistemas detección de incendios que funcionan exclusivamente con tecnologías LPWAN permitiendo una gran área de cobertura y un consumo bajo.

Los sistemas comerciales, como por ejemplo los sistemas de alarmas domésticos, utilizan sistemas de comunicación con infraestructuras profundamente implantadas, como lo son las redes móviles. Este tipo de redes permiten una gran cobertura y una tasa de transmisión alta que permite el envío de imágenes de video y audio a la central, pero a un coste energético muy alto, lo que obliga al Gateway estar cableado a la red eléctrica.

En la propuesta de este trabajo se quiere ir un paso más allá, y se pretende desarrollar un sistema hibrido muy parecido al propuesto en el ejemplo 1 pero añadiendo otra tecnología más en la arquitectura, en este caso Sigfox.

En la siguiente imagen se puede observar las principales ventajas que aporta cada una de las tecnologías propuestas para la construcción del sistema.

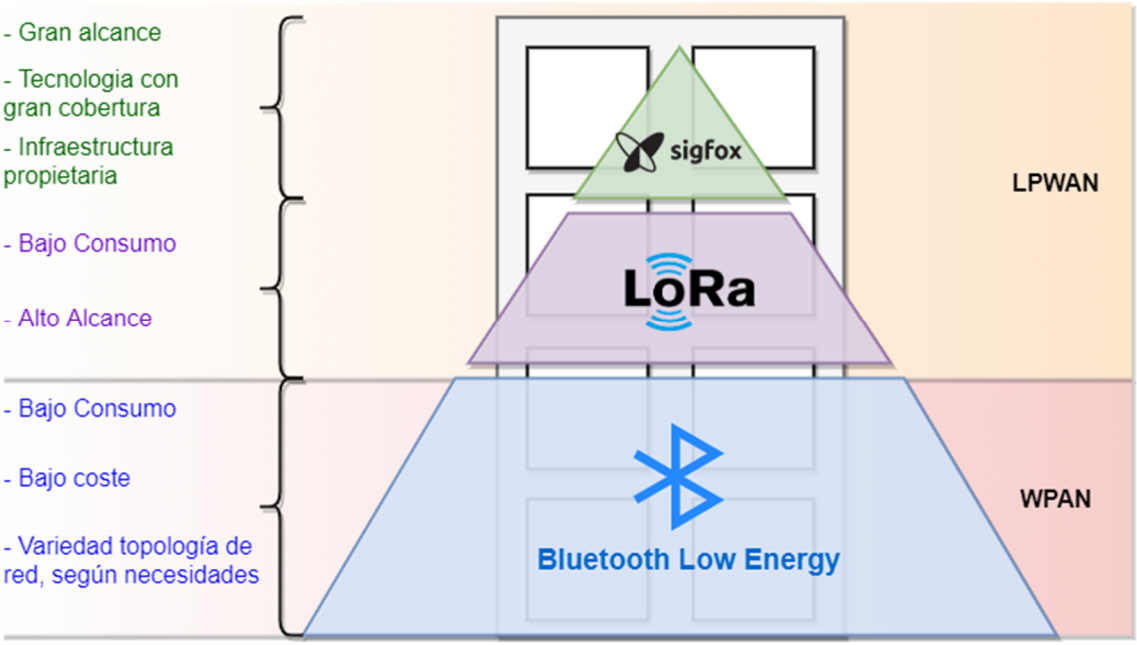



*Figura 12: Principales aportaciones de cada sistema de comunicación a la arquitectura.*

- **Sigfox,** es la tecnología encargada de conectar al sistema con la nube, recogerá todas las tramas recibidas por los gateway LoRa y los empaquetará para enviarlos al servidor propietario.

  Aportará al sistema**:**

  - **Gran alcance**: A diferencia que en los anteriores sistemas descritos, como LoRa o NB-IoT [27], Sigfox como tecnología de enlace encargada de la comunicación con la nube y como puente a una red externa, nos permite un gran alcance que el resto de las tecnologías LPWAN no son capaces de igualar.
    También como se ha mostrado en el ejemplo 4, existen tecnologías con cobertura global como por ejemplo las redes WAN (*Wide Area Network*), pero la movilidad de estos sistemas como la posibilidad de posicionamiento en lugares desatendidos, es nula y para un sistema de emergencias es mejor el uso de una red menos compleja y una topología con una mayor robustez ante cortes (tanto eléctricos como de datos) como lo puede ser una red LPWAN como Sigfox en la cual si falla una estación permite la conexión a otra diferente.

  - **Tecnología con gran cobertura**: Sigfox se trata de una tecnología que dispone de una gran cobertura a nivel europeo [28] En otras tecnologías como por ejemplo LoRaWAN la cobertura en el mismo area geográfica es inferior.

  - **Infraestructura propietaria**: Al tratarse de una tecnología propietaria con su propia infraestructura de comunicación, el coste inicial derivado de este ámbito es muy inferior y solamente se tendrían que costear los servicios asociados a la suscripción anual de cada dispositivo. En otros sistemas sería necesario realizar una gran inversión inicial para la implantación de la infraestructura de comunicación.
    Si lo comparamos a otros sistemas basados en servicios con suscripción como NB-IoT y redes como fibra óptica o redes de Telefonía móvil también hay que asumir un coste mensual, mayor que el coste del servicio de Sigfox.

- **LoRa,** será la tecnología de comunicación encargada de transmitir la información procedente de cada subred WPAN al gateway Sigfox.

  Sus aportes al sistema son**:**

  - **Bajo consumo**: El consumo de un sistema de comunicaciones por LoRa es muy bajo respecto a otras tecnologías, permitiendo una gran autonomía.

  - **Largo alcance:** El alcance de estos sistemas permite comunicaciones de hasta 5km, en el caso del sistema propuesto permitirá la comunicación entre ambos niveles de gateway a pesar de los obstáculos naturales situados en el interior del edificio.

- **Bluetooth Low Energy,** tecnología de comunicación empleada para la comunicación entre los nodos-sensores y el gateway de la vivienda.

  Sus aportes al sistema son:

  - **Bajo consumo:** La tecnología bluetooth low energy está orientada a comunicaciones IoT, por lo tanto, esta optimizado su consumo energético, permitiendo comunicaciones inalámbricas con una alta eficiencia energética.

  - **Bajo coste:** La gran mayoría de las unidades de procesamiento comerciales orientadas a IoT disponen por defecto de transceptores bluetooth integrados en los propios dispositivos. Por lo tanto, el uso de esta tecnología de comunicación inalámbrica permite una gran variedad de dispositivos comerciales donde elegir y a un coste muy competitivo. En cambio, en otro tipo de tecnologías de comunicación como, por ejemplo, el uso de comunicaciones LPWAN directamente sobre los dispositivos sensores (Ejemplo 2 y 3), es necesaria la adquisición de transceptores externos para poder realizar las comunicaciones, además de la electrónica complementaria necesaria para su funcionamiento, lo que incrementa el coste de fabricación del dispositivo y por tanto el precio final del sistema.

  - **Variedad de topologías de red, según necesidades**: Bluetooth permite el uso de diferentes topologías de red como punto a punto o estrella como topologías más convencionales. Aunque actualmente la tendencia del mercado IoT está muy enfocada hacia una topología de red muy interesante, la topología de red en malla o *mesh,* la cual permite a cada nodo actuar como Gateway de otros dispositivos de la red por lo que en este caso no es necesaria la visibilidad del nodo con el dispositivo aprovisionador del servicio. Una de las ventajas que permite esta configuración es un aumento de la cobertura de la red más allá de lo posible normalmente por una red WPAN.

La siguiente tabla es una comparativa de los datos más característicos de cada tecnología en las arquitecturas analizadas con respecto al sistema propuesto

| | **Sistema 1: Mobile 2019** | **Sistema 2: Semtech** | **Sistema 3: Sistema comercial SiterWell** | **Sistema 4: WSN IEEE 2016** | **Sistema 5: Alarma comercial** | **Modelo propuesto** |
|---|---|---|---|---|---|---|
| **Tecnología Nodo** | WPAN: Bluetooth Low Energy | LPWAN: LoRa | LPWAN: NB-IoT, Sigfox, LoRa | WLAN: WiFi | WPAN/RF propietario | WPAN: Bluetooth Low Energy |
| **Tecnología Gateway** | LPWAN: LoRa | LPWAN: LoRa | LPWAN: NB-IoT, Sigfox, LoRa | WAN: Fibra/ADSL | Celular: 2G/3G | LPWAN: LoRa(L1) Sigfox(L2) |
| **Frecuencia (Nodo)** | 2.4 GHz | ISM 433MHz/ 868 MHz/ <2GHz | ISM 433MHz/ 868 MHz/ <2GHz | <GHz, 2,4GHz, 5GHz | <GHz, | 2.4 GHz |
| **Frecuencia (Gateway)** | ISM 433MHz/ 868 MHz | ISM 433MHz/ 868 MHz | ISM 433MHz/ 868 MHz/ Celular. <2GHz | <GHz, 2,4GHz, 5GHz | Celular. <2GHz | ISM 433MHz/ 868 MHz/ |
| **Tasa de transferencia (Nodo)** | 1,2,3 Mbps | <50Kbps | LoRa: <50kbps Sigfox: <1Kbps NB-IoT< 200kbps | 802.11ac < 866 Mbps<br><br>802.11b < 11 Mbps | < 1 Gbps | 1Mbps |
| **Tasa de transferencia (Gateway)** | <50Kbps | <50Kbps | LoRa: <50kbps Sigfox: <1Kbps NB-IoT< 200kbps | <1 Gbps | <10 Mbps | LoRa: <50kbps Sigfox: <1Kbps |
| **Rango (Nodo)** | 10-100 m | 5-15km | LoRa:5-15Km SigFox:5-50Km NB-IoT:1-10 Km | 10-100m | 10-100m | 10-100m entre nodos |
| **Rango (Gateway)** | 5-15km | 5-15km | LoRa:5-15Km SigFox:5-50Km NB-IoT: 1-10 Km | Global | <35 km | LoRa:5-15Km SigFox:5-50Km |
| **Consumo (Nodo)** | Bajo | Muy Bajo | Muy Bajo | Medio | Bajo-Medio | Muy Bajo |
| **Consumo (Gateway)** | Muy Bajo | Muy Bajo | Muy Bajo | Alto | Alto | Muy Bajo |
| **Coste (Dispositivos)** | Muy Bajo | Bajo | Bajo | Bajo | Medio | Bajo |
| **Coste (Servicio)** | Ninguno | Ninguno | Sigfox (25€/año/Gate) Lora: Ninguno NB-IoT: 500 millones/ MHz | Si (30€/mes/Gate) | Si (10€/mes/Gate) | Sigfox (16€/año/Gate) |
| **Coste (Infraestructura)** | Alto | Alto | Sigfox, Ninguno LoRa: Alto NB-IoT: Bajo | Ninguno | Ninguno | Ninguno |

*Tabla 3: Comparativa de tecnologías usadas en sistemas analizados* [29][30][31][32]

# 3. Fundamentos teóricos de sistema de redes de sensores.

A continuación, se expondrá de brevemente una introducción a los sistemas de redes de sensores, infraestructura en la que se basa el sistema de este trabajo.

## 3.1 Introducción

Una red de sensores se trata de una infraestructura de comunicación formada por un conjunto de dispositivos que detecta cualquier cambio en el entorno a través de sensores. Este grupo de dispositivos recopila y almacena dicha información transmitiéndola típicamente de manera inalámbrica y distribuida a través de los nodos que la constituyen. Se caracterizan por ser dispositivos de pequeño tamaño, de composición física sencilla, tener un bajo coste y un consumo reducido.

El escenario de uso de este tipo de redes tiene una serie de restricciones y características sobre las redes ad hoc:

- Están pensados para poder ser colocados en lugares poco accesibles, deben comunicarse de forma inalámbrica.
- Deben ser sistemas distribuidos, con una arquitectura de red cambiante, por lo que cada nodo debe adaptarse. Esto se puede realizar a través de capacidades de autoorganización en la que se incluye también una capacidad de colaboración entre nodos para enrutamiento y envío de información cuando otro tipo de topologías no permiten la comunicación de un nodo con el destino final.
- Están limitados en consumo, tamaño, coste, capacidad de procesamiento y cantidad de memoria.
- No tienen una identificación global debido a la gran densidad de dispositivos.

## 3.2 Características

Este tipo de redes y los nodos que las forman se enfrentan una serie de retos y dificultades que hace que deba cumplir una serie de características **[33]:**

- **Coste:** La red está formada por una gran cantidad de nodos, el coste de producción de estos es importante y por lo tanto, el coste por unidad debe ser bajo para justificar el coste total de la red respecto a otro tipo de soluciones tradicionales

- **Eficiencia energética/ Consumo:** La cantidad de energía disponible en un dispositivo es muy limitado y solo dispondrá de la cantidad de energía disponible en su batería, por lo que el consumo del dispositivo debe ser muy bajo en todo momento. También en su unidad de alimentación puede disponer de un método de generación de energía que le permita obtener una capacidad de autonomía eléctrica continua.

- **Escalabilidad** Las redes de sensores pueden estar formadas por cientos o miles de dispositivos, además puede llegar a tener una gran cantidad densidad en un área determinada, por lo tanto, debe ser un sistema escalable que permita el aumento de su tamaño a un coste proporcionalmente lineal y su funcionamiento debe ser similar en todas las dimensiones que pueda adoptar la red.

- **Flexibilidad y topología de red:** La red de sensores debe ser un sistema flexible que pueda adaptarse ante cambios en la infraestructura, como eliminación de nodos, despliegue de más unidades, actualizaciones, o posibles fallos en los nodos que obliguen a un cambio en el canal de comunicación entre nodos.

- **HW / Tolerancia a fallos.** Los nodos deben estar diseñados de una manera simple, ya que están limitados por el consumo de energía. Están formados por 4 bloques básicos: unidad de procesamiento, unidad de alimentación, unidad de medición y unidad de transceptor, todas estas unidades deben ser controlables según la capacidad de energía disponible. También cada dispositivo debe ser robusto y tener un ciclo de vida elevado, para evitar en la medida de lo posible la sustitución de sus componentes de una manera frecuente.

- **Transmisión/recepción:** Respecto al acceso al medio debe realizarse de forma inalámbrica, mediante el uso de sistemas de comunicación orientados a bajo consumo, radio, infrarrojos, óptico, etc. También la red debe tener cierta adaptabilidad en los canales de transmisión y recepción entre nodos, debido a interferencias ambientales, apagados de dispositivos, fallos, falta de visión directa entre dispositivos, etc.

- **Programable:** La reprogramación de la unidad de control de cada nodo debe ser posible, ya que podría ser necesaria para actualizaciones que permitan una mejora de eficiencia y flexibilidad de la red.

- **Seguridad [34]:** El nodo debe ser un sistema seguro, para ello debe cumplir una serie de características respecto a este factor:
    - Control de acceso: evitar accesos no autorizados a la red y al nodo.
    - Integridad de mensaje: detección y prevenir cambios no autorizados en el mensaje.
    - Confidencialidad: El nodo debe encriptar los mensajes para que solo aquellos nodos que tengan la clave puedan escucharlo.
    - Protección: compresión de envió de datos en ventanas de tiempo para evitar la escucha por parte de intrusos.

### 3.3 Arquitectura de un nodo/sensor.

La estructura básica de un sensor se puede dividir en cuatro bloques principalmente:

- **Unidad de alimentación:** Formado por un circuito electrónico de regulación de energía, así como de un gestor de batería y un sistema de almacenamiento o batería. El bloque se encargará de proveer al resto de sistemas de energía eléctrica adaptada para su funcionamiento. Cuando esté conectado a la red eléctrica deberá mantener su funcionamiento y a la vez recargar la unidad de almacenamiento.

- **Unidad de comunicación:** Esta constituido principalmente por el dispositivo transmisor/receptor al medio de comunicación del sensor. Este permitirá la comunicación entre los diferentes integrantes de la red.

- **Unidad de procesamiento:** Bloque encargado de recolectar, procesar, almacenar y fusionar la información procedente de la unidad sensor. Está formado por un microprocesador y una memoria. La tendencia del mercado actual es que tanto el bloque de comunicación como el de procesamiento se encuentren en un mismo circuito integrado formando unidades SoC (*System on Chip*).

- **Unidad de medición:** Bloque encargado de la medición de distintos parámetros externos del entorno que le rodea, como puede ser temperatura, humedad, presión, luz, etc. Los valores medidos pueden ser ofrecidos directamente en formato digital o en una señal analógica. En este último caso, sería necesario convertirlos a un formato digital para ser legibles por la unidad de procesamiento por lo que es necesario que añadir un ADC (*Analogic Digital Converter*).

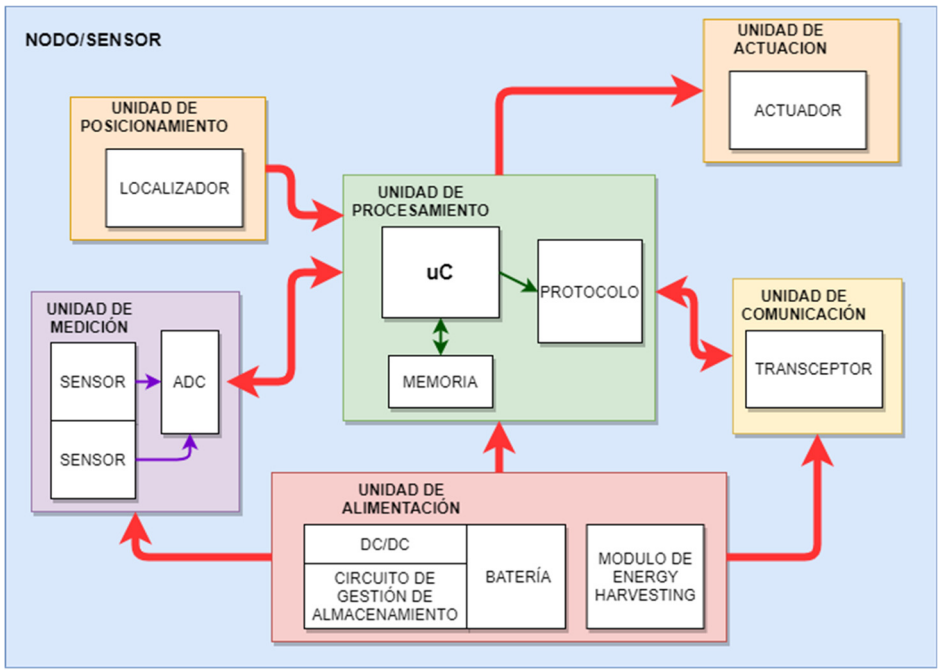



*Figura 13: Diagrama genérico de la arquitectura de un nodo/sensor* [35]

Como se puede observar en la figura 13, se han incluido dos bloques que no están disponibles en todos los nodos y se pueden considerar opcionales, según la función principal de la red. Entre ellos se pueden encontrar:

- **Unidad de actuación:** se puede considerar una parte de la unidad de procesamiento. Según los parámetros recolectados a través de la unidad de medición o las órdenes recibidas a través de la unidad de comunicación, la unidad de procesamiento manda una orden al bloque actuador para modificar su estado, activando o desactivándolo, provocando una respuesta a la medida.

- **Unidad de posicionamiento:** Existen sensores que incluyen este bloque. Se le podría considerar un componente de la unidad de medición, con la diferencia que este sirve para conocer la posición geográfica o relativa del dispositivo.

# 4. Desarrollo de PCB modular con conectividad BLE, WiFi y LoRa.

El diseño de sistemas electrónicos y placas de circuito impreso es un proceso complejo que se puede dividir en varias etapas:

- Planteamiento de un problema
- Búsqueda de información
- Diseño de una solución
- Diseño y desarrollo de esquemático
- Rutado de placa.
- Documentación
- Montaje
- Puesta en marcha

En el siguiente capítulo se explicará estos procesos para el diseño y desarrollo de un sistema electrónico modular orientado a cubrir las necesidades presentadas en el proyecto. Dicho sistema dispondrá de varios módulos intercambiables a modo de periféricos según la funcionalidad a la que vaya destinado el dispositivo.

## 4.1 Sistema modular. Descripción y especificaciones técnicas.

Como se ha podido ver en apartados anteriores, en el sistema de emergencias propuesto intervienen varios tipos de dispositivos. Estos son:

- Dispositivo nodo/sensor: para tomar medidas y enviar a través de BLE la información referente a la alarma disparada.
- Gateway BLE con conectividad LPWAN, LoRa
- Gateway LoRa con conectividad LPWAN Sigfox con acceso a backend y a red

Debido a que el último de ellos tiene conectividad con una tecnología propietaria, hace que el diseño de un sistema con este tipo de conectividad sea muy complejo, ya que sería necesario el estudio del protocolo o el uso de alguno de los componentes propuestos por el fabricante.
Para los dos primeros se propone el diseño de una electrónica, con varios módulos conectables. Según la funcionalidad objetivo del dispositivo deberán conectarse unos u otros.

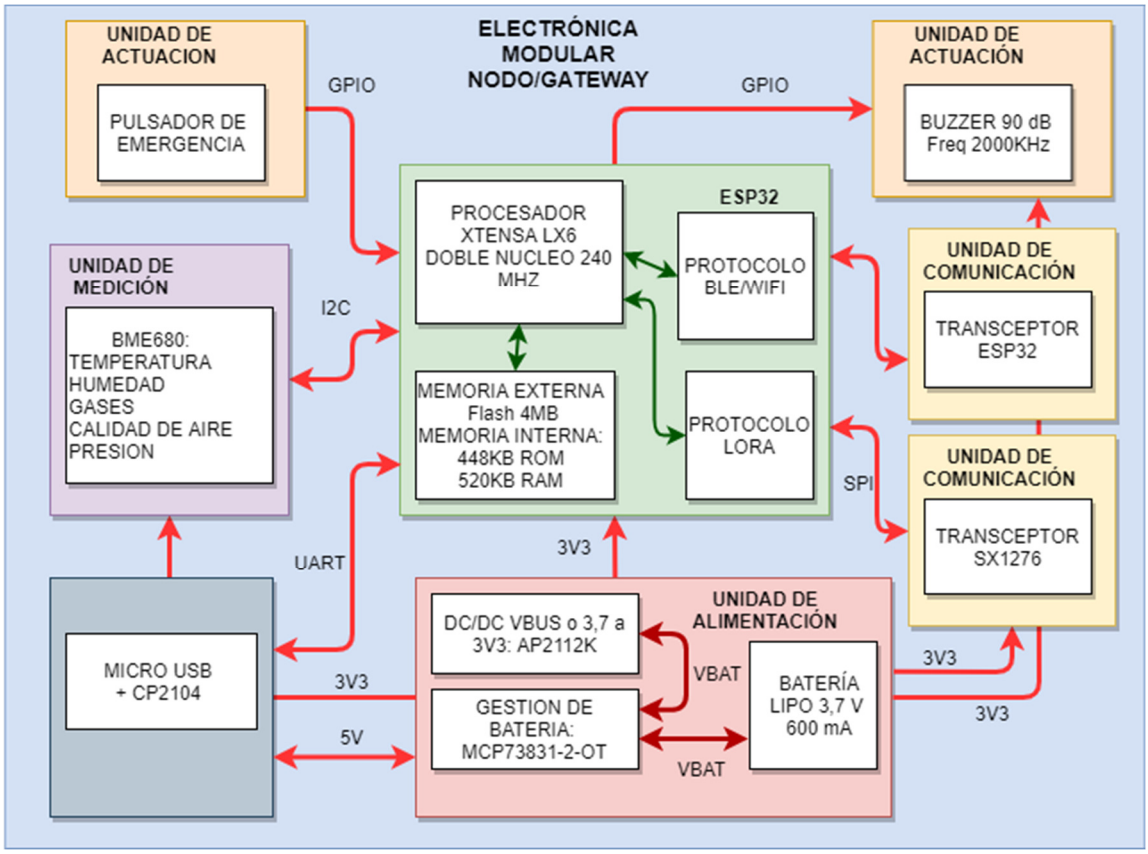



*Figura 14: Diagrama de sistema electrónico modular ESP32*
*Diagrama basado en la clasificación expuesta en el capítulo 3 del documento. La placa puede funcionar como dos sistemas diferentes, como nodo o como Gateway LoRa, según su función usará unos bloques u otros*

A continuación, se describirá el proceso de selección de cada uno de los módulos, así como el integrado finalmente elegido:

- **Unidad de procesamiento (nodo/gateway):**
  Opciones propuestas:
  - STM32WB de ST [36]
  - ESP32 de Espressif [37]
  - nRF52832 de NORDIC semiconductor [38].

  La elección de la unidad de procesamiento fue bastante compleja ya que existen muchos factores diferenciadores entre las opciones. En este caso, se trata de tres dispositivos muy diferentes. El modelo ESP32 es un microcontrolador muy potente, pero tiene un consumo más elevado que las otras dos opciones. Por otro lado, es un integrado que puede comprarse en forma SoM (*System on Module*) y viene con todo lo necesario para su funcionamiento sin la necesidad de comprar componentes externos como cristales u osciladores para su funcionamiento. Dispone de más cantidad de almacenamiento y también de conectividad WiFi que permitiría actualización de software OTA (*Over The Air*).

Respecto a las otras dos opciones, se trata de dispositivos muy enfocados a las comunicaciones inalámbricas, su consumo es realmente bajo, tanto de forma activa como en modo de bajo consumo. Por el contrario, son sistemas que requieren de gran cantidad de componentes externos, disponen de poca memoria para el almacenamiento de programa y datos, no están tan extendidos entre la comunidad de desarrolladores, no disponen de conectividad WiFi y su coste en general es mucho más elevado que la solución con el ESP32. Todos estos factores inclinaron la balanza hacia el ESP32.
Finalmente, el módulo que se ha optado ha sido el modelo TTGO Micro-32 de la marca Lilygo. Este módulo está basado en un microcontrolador ESP32 PICO D4 [39] el cual dispone ya de todos los componentes básicos para su funcionamiento y comunicación, como transceptor BLE y WiFi, filtro de entrada, cristal de forma nativa, incluyendo una antena cerámica de reducido tamaño. Además, esta soldado sobre un soporte PCB para facilitar la soldadura en prototipos.

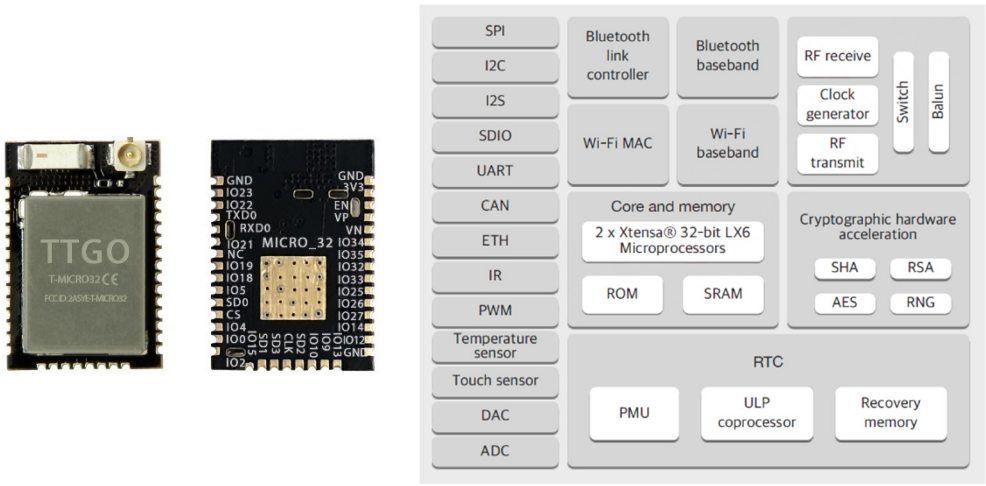



*Figura 15: modulo micro-32 TTGO/diagrama interno de ESP32*

Respecto a sus especificaciones técnicas, el núcleo de este dispositivo se basa en un procesador Xtensa LX6 doble núcleo a 240 MHz aportándole una velocidad de computación de 600 DMIPS (*Dhrystone Million Instructions Per Second*). Dispone de un módulo de memoria externa flash con una capacidad de 4MB además de la memoria interna, la cual contiene 448KB de memoria ROM (*Read Only Memory*) destinada a programa y 520KB de memoria RAM (*Random Access Memory*) para datos.
La principal ventaja de este microcontrolador es la gran cantidad de apoyo y recursos disponibles en internet, tanto por parte de la comunidad como por parte del fabricante. Además, permite varias opciones de lenguajes de programación, inclusive MicroPython, lo que supone una gran opción para realizar prototipos de forma rápida y sencilla.
Respecto al consumo del sistema, este modelo de microcontrolador dispone de varios modos de funcionamiento, cada uno de ellos con un consumo diferente gracias al apagado de los periféricos disponibles.

| Modo | Descripción | Consumo |
|---|---|---|
| **Activo (RF+CPU)** | Transceptor WiFi/BT (*bluetooth*) encendido y trabajando en la recepción o transmisión | Tx WiFi: 180 -240 mA |
| | | Rx WiFi: 95-100 mA |
| | | Tx BT/BLE: 130mA |
| | | Rx BT/BLE: 95-100 mA |
| **Modem-Sleep** | CPU encendida, pero elementos de comunicación inalámbrica apagados. | CPU 240MHz:30-50mA |
| | | CPU 80MHz:20-25mA |
| | | CPU 2MHz:2mA-5mA |
| **Light-Sleep** | CPU pausada y cuando se reestablezca el sistema retomará la última tarea, módulos de comunicación inalámbrica apagados. Periféricos y memorias RTC y procesador ULP encendido. | 0,8 mA |
| **Deep-Sleep** | CPU y transceptores apagados, ULP y memoria RTC pueden funcionar, se puede salir de este estado por métodos externos GPIO (*General purpose input/output*) o internos (Timer) | ULP on: 150 uA |
| | | Monitorizar sensor 1% por ULP: 100 uA |
| | | RTC timer + RTC memoria: 10uA |

*Tabla 4: Modos de funcionamiento ESP32 y consumos.*

El uso de estos modos de funcionamiento a la hora del desarrollo del firmware de configuración del microcontrolador será clave a la hora de conseguir un sistema eficiente energéticamente y con una autonomía elevada.

- **Unidad de comunicación 1 (nodo/gateway):** Transceptor WiFi/BLE.
  De forma nativa el microcontrolador ESP32 incluye un transceptor orientado para conexión en la banda de 2,4 GHz. Entre los que se encuentran la tecnología WiFi y bluetooth 4.2 con posibilidad de bluetooth low energy. Este módulo será utilizado en el sistema para la comunicación de los nodos con el Gateway de la vivienda, formando una red WPAN que dará cobertura ante emergencias en toda la vivienda.
  Debido a su alto consumo en funcionamiento, la mayor parte del tiempo deberá estar apagado para ahorrar energía.

- **Unidad de comunicación 2(Gateway):**
  Opciones propuestas: Todos los módulos existentes se basan en tecnología desarrollada por Semtech.
  Ejemplos:
  - CMWX1ZZABZ-078 **[40]**
  - RM186-SM-01 **[41].**

  Ambos modelos son módulos con microcontrolador que están basados en transceptores de Semtech. Al ser una solución cerrada y con unidad de procesamiento incluida, su coste económico es más elevado. Otra desventaja es que al disponer de herramientas propietarias limita las opciones de programación del desarrollador. Esto llevo a la conclusión de

incluir un módulo transceptor con únicamente el transceptor LoRa de Semtech.
Este módulo trabaja en la frecuencia ISM 868 MHz y estará orientado para la comunicación a través de LoRa de la información procedente de la subred WPAN con el Gateway Sigfox.

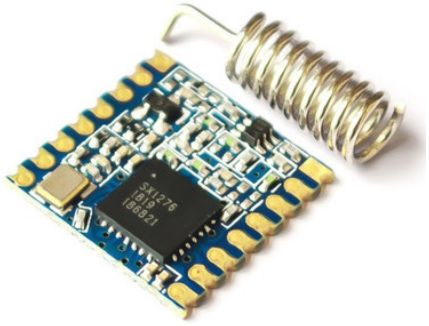


*Figura 16: Modulo transceptor ISM 868 MHz basado en chip Semtech SX1276* [42]

El transceptor que contiene el módulo es el modelo SX1276 **[42]** de Semtech Adicionalmente mediante diseño PCB, este módulo estará conectado a una antena cerámica para reducir al máximo el tamaño del dispositivo. De ser necesario un mayor alcance, se podría conectar una antena de mayor ganancia a través del conector ULF y soldando una resistencia a la entrada de la señal para cambiar la selección de antena. La comunicación entre el transceptor y el microcontrolador ESP32 se realizará a través del protocolo SPI.
Respecto a su consumo, debido a su elevado requerimiento de energía y a que solamente se utilizará en momentos puntuales, este módulo deberá estar la mayor parte del tiempo apagado, y activarlo solamente en caso de transmisión de datos.

- **Unidad de medición: Sensor de temperatura/humedad/gases. (Nodo**)
  Opciones propuestas:
  - Conjunto de sensores formado por CCS811**[43]** + HDC1080 **[44]**.
  - BME680 de Bosch **[45]**.

Ambas son buenas opciones y con precio similar. La opción BME680 finalmente fue elegia por su mayor rango de temperatura de operación, mejor resolución y menor consumo: La solución basada en CCS811 + HDC1080 era muy atractiva pero su disponibilidad en el mercado no era muy elevada, además de ser necesarios dos componentes para cumplir la misma función que el sensor de Bosch.

Por lo tanto, el elemento principal y único para recolección de parámetros estará compuesto el modelo BME680. Este sensor es un dispositivo orientado a aplicaciones ambientales, donde es muy importante el tamaño y el consumo de energía limitado. Por lo que para el diseño propuesto para el presente trabajo será muy importante a la hora de conseguir una autonomía elevada. Dispone de varias opciones de comunicación, tales como SPI (*Serial Peripherical Interface*) o I2C (*Inter-Integrated Circuit*), en el caso del diseño actual se ha utilizado I2C para reducir el número de

pines necesarios para su configuración. La función principal será la de medición de la temperatura ambiental del habitáculo donde este establecido el nodo, además de realizar pruebas en la medición de gases para ver si la respuesta de este parámetro también es correcta pudiéndose utilizar como un parámetro complementario. Este dispositivo es sensible ante Etano, monóxido de carbono, Acetona, Etanol Isopropanol y Butano, y aunque no es capaz de dar un valor cuantitativo de cada uno de ellos, mediante una función y algoritmos propietarios es capaz de indicarnos unos rangos de concentración de gases denominado IAQ (*Indoor Air Quality*, Tabla 6)

*Figura 17: Encapsulado de sensor Bosch BME680* **[45]**

| **BME680** | |
|---|---|
| Interfaz | I2C o SPI |
| Resolución | Temperatura: 1ºC<br>Humedad: 3%<br>Gases: 5%<br>Presion:0,6 HPa |
| Consumo | 2,1uA 1 Hz Humedad y Temperatura<br>3,1uA 1Hz Presión y temperatura<br>3,7uA 1Hz H, P y T<br>0,09-12 mA Gases<br>0,15uA En modo sleep |
| Voltaje alimentación | 1,7V-3,6V |

*Tabla 5: Características más importantes de BME680* **[45].**

| IAQ Index | Air Quality | Impact (long-term exposure) | Suggested action |
|---|---|---|---|
| 0 – 50 | Excellent | Pure air; best for well-being | No measures needed |
| 51 – 100 | Good | No irritation or impact on well-being | No measures needed |
| 101 – 150 | Lightly polluted | Reduction of well-being possible | Ventilation suggested |
| 151 – 200 | Moderately polluted | More significant irritation possible | Increase ventilation with clean air |
| 201 – 250[9] | Heavily polluted | Exposition might lead to effects like headache depending on type of VOCs | optimize ventilation |
| 251 – 350 | Severely polluted | More severe health issue possible if harmful VOC present | Contamination should be identified if level is reached even w/o presence of people; maximize ventilation & reduce attendance |
| > 351 | Extremely polluted | Headaches, additional neurotoxic effects possible | Contamination needs to be identified; avoid presence in room and maximize ventilation |

*Tabla 6: Tabla cuantificativa de parámetro IAQ con el sensor BME680* **[45].**
*Representa los diferentes niveles representativos de calidad del aire cuantificados por el valor de IAQ, así como acciones a tomar o posibles efectos en caso de alcanzar a dicho nivel*

- **Unidad de actuación 1: Alarma sonora (Nodo**).
  Opciones propuestas:
  - Altavoz de membrana
  - Buzzer.

  Finalmente, de tomo la decisión de integrar un Buzzer debido a su baja complejidad, bajos requerimientos exteriores y alto valor de ganancia acústica de emisión. La solución de un altavoz requería del uso de una etapa de amplificación que implicaba un mayor coste.
  El módulo por diseñar se trata de una placa periférica, que dispondrá de un pequeño zumbador [46] acústico a 2KHz además de la electrónica necesaria para su correcto funcionamiento. Con solamente 3 pines, 2 de alimentación y uno de control, con una señal cuadrada modulada a su frecuencia de resonancia, permite emitir un sonido de hasta 90 dB.

*Figura 18: Encapsulado típico de Buzzer de inserción*

- **Unidad de actuación 2: Pulsador de emergencia [47] (Nodo).**
  Se trata de una unidad incorporada directamente en la placa principal que dispone de un pulsador. Este estará conectado directamente a la unidad de procesamiento. Cuando en dicho pulsador se realice un número determinado de pulsaciones seguidas, se considerará un estado de alarma a pesar de que ninguno de los parámetros medidos por el sensor haya superado ningún umbral. Será especialmente útil para emergencias médicas.

- **Unidad de alimentación: Regulador + Gestor de batería (Nodo/Gateway).**
  En este caso, disponemos de varias opciones de alimentación que previamente deberán ser sometidas a pruebas a lo largo del desarrollo practico del proyecto. La elección de cada una de ellas dependerá de la autonomía máxima conseguida o para que dispositivo este orientado.

  - **Opción 1: Alimentación con batería recargables de litio o polímero de litio 3,7V**
    Esta opción está formada por el regulador lineal LDO (*Low Drop Out*) *step-down* AP2112K [48], el gestor de batería MCP73831/2 [49] y una batería de litio LiPo (*Litio Polimerer*) (principalmente por su tamaño y forma). El regulador lineal se encargará de convertir el voltaje a su entrada a 3V3, para ello siempre es necesario que dicho voltaje sea mayor al de su salida. Esta unidad también dispondrá de un selector de alimentaciones formado por un diodo

Schottky según la situación del puerto micro USB de la placa. Si esta se encuentra conectada y alimentada por 5V, el regulador funcionará con la alimentación VBUS procedente del conector, además el circuito que gestiona la alimentación de la batería se dispondrá a la carga de esta. En cambio, si no existe ninguna alimentación en VBUS, el circuito recurrirá a la energía almacenada en la batería para el funcionamiento del resto del circuito. La principal ventaja de esta opción es que se puede recargar la batería mediante la conexión a la red a través de un AC/DC a 5V a USB. Por el contrario, debido a que el rango de tensión del circuito es como máximo hasta 3,6V no podemos conectar directamente la batería a la alimentación general, y es necesario el uso de un circuito de regulación de tensión. Este tipo de circuitos tienen perdidas que son muy relevantes si las comparamos con los consumos del circuito en Deep sleep (orden de uA) y por lo tanto no será adecuada su uso para productos finales, aunque si para sistemas cableados o para el desarrollo de prototipos.

- **Opción 2: Alimentación con pila de litio.**
  Existen pilas de litio con voltaje de 1,5V o pilas de formato CR123 a 3V, que, a un valor de 2,55V las pilas han aportado el mayor porcentaje de carga y están prácticamente vacías, y como la tensión mínima de ESP32 es 2,3V se podría aprovechar al máximo su capacidad de alimentación. Por otro lado, su rango de voltaje se sitúa en el rango de funcionamiento del circuito, por lo que no será necesaria ninguna etapa de regulación y por lo tanto no tendrá ninguna perdida en esta etapa. Por el contrario, este tipo de pilas no son recargables y por lo tanto es necesario un mantenimiento cada cierto periodo de tiempo. Esta solución será ideal para un producto finalizado y ya verificado el funcionamiento del circuito en los rangos de consumo propuestos para producción.

- **Otros (Nodo/Gateway).**
  El circuito también contiene aparte de un conector micro USB para la carga y alimentación del circuito, de un conversor USB a puerto serie, el modelo CP2104 **[50]**. Este integrado es esencial para la programación del dispositivo y para realizar operaciones de depuración del firmware a través del puerto serie.

### 4.2 Esquema

El esquema de un circuito es una representación simbólica de un circuito electrónico. Muestra los diferentes componentes de un modo simplificado para facilitar la compresión y lectura por el usuario del funcionamiento del circuito.
Todo el diseño de la placa de circuito impreso (PCB) se ha realizado a través de KiCAD **[51],** un software de diseño electrónico de libre distribución. La parte de diseño esquemático se caracteriza principalmente por seguir una estructura jerárquica donde el diagrama completo del esquema se puede observar en un alto nivel en la página principal y luego acceder a cada uno de los esquemas de

bajo nivel para obtener un mayor detalle de cada uno de los circuitos. La estructura jerárquica del esquemático es la siguiente:

**Home.sch:** página principal contiene los bloques inferiores, en esta página se puede observar la estructura principal del circuito. Está compuesto por 3 bloques:

- **Power.sch:** Bloque compuesto por la unidad de alimentación, la etapa de entrada micro USB y el conversor USB a UART.

- **Micro.sch:** Contiene la unidad de procesamiento, así como la electrónica necesaria para reiniciar y cambiar el modo de arranque del microcontrolador.

- **Perifericals.sch**: Contiene todos los módulos periféricos disponibles para conectar a la placa principal. En ellos podemos encontrar la placa Buzzer, la placa de medición y la placa de comunicación a través de LoRa.

Después del desarrollo del esquemático, anotación de referencias y chequeo de los esquemas, se procede con la asociación de huellas a los símbolos del circuito. KiCAD tiene la gran ventaja de que viene con una extensa librería de componentes por lo que en la mayoría de los símbolos no es necesario el diseño de una huella.
Finalmente, tras la asociación de huellas, se puede generar el fichero netlist y así proceder con el rutado de la PCB.

*El esquema completo del diseño se puede encontrar en el ANEXO A de la memoria del trabajo.*

### 4.3 Rutado

El rutado de la placa es el proceso en el que los componentes propuestos en el esquema se disponen en una posición física en la placa de circuito impreso. Para ello es necesaria la carga de las librerías de componentes que se van a utilizar en el PCB, así como la netlist generada desde la aplicación de esquemáticos. Esta lista contiene toda la información relativa a la interconexión de los componentes de la placa.
Pero antes de empezar con el agrupamiento de componentes y rutado de pistas, es necesario el estudio y una toma de una serie decisiones respecto a las características físicas de la placa. Este punto es de vital importancia para el correcto desarrollo de un dispositivo inicialmente orientado a la producción masiva, aún más si cabe tratándose de un sistema RF. Dichas características dependerán de una serie de factores:

- **Numero de capas:** El número de capas de cobre debe ser un número par y el contenido de cada una de ellas tiene que ser lo más simétrico posible para evitar torsiones de la propia placa **[52] [53].**
  Su número dependerá de:
  - El número de pistas y alimentaciones necesarias en la placa.
  - La necesidad de control de impedancias en las pistas, sobre todo en señales de alta velocidad.

- **Separación entre cobre.** La distancia física entre pistas o vías depende de:
  - La capacidad tecnológica del fabricante seleccionado para la fabricación del PCB.
  - El paso de pad (distancia entre pads) de los componentes BGA (*Ball Grid Array*)
  - La separación entre señales diferenciales y la distancia de estas con el resto de las señales que puedan intervenir y provocar diafonía o *crosstalk* sobre las señales sensibles.

- **Ancho de pista y vía**: El ancho de pista dependerá del tipo de señal que viaje por ella, por ejemplo, dentro de una placa de circuito impreso podemos encontrar señales sensibles de alta velocidad, como señales de video, señales de control a baja frecuencia, como reset, GPIO (*General purpose input/output*), pistas de alimentación en las que circula un voltaje con un elevado flujo de corriente, u otro tipo de señales analógicas o digitales. Según el tipo de señal, la corriente que circule por ella y la impedancia necesaria para evitar pérdidas de integridad deberá tener un ancho especifico y una distancia determinada con su plano de referencia. También dependerá de la capacidad del fabricante y del tipo de prepreg (capa de dieléctrico entre capas de cobre) que disponga el fabricante.

- **Terminación de superficie**: se refiere a los diferentes baños de material a los que puede estar sometida la placa. Esta terminación dependerá de las normativas a las que esté sujeta (marcados ROHS (*Restriction of Hazardous Substances*)) o de las condiciones a las que esté sometido el dispositivo, por ejemplo, un dispositivo orientado a un funcionamiento en exteriores estará sometido a un mayor rango de temperaturas y a una determinada humedad, lo que facilita la oxidación de sus conexiones y aumente el riesgo de fallo de la placa.
  Los tipos más comunes de terminación son: HASL (*hot air solder leveling*), ENIG (*Electroless Nickel/ Immersion Gold*) y IAg ( *Immersion Silver Plating*)

- **Material:** El tipo de material dieléctrico utilizado para la construcción de la PCB. Existe gran variedad FR-1,2,3,4, teflón, cerámica o CEM-1,2,3,4,5. Su elección dependerá de la constante dieléctrica, temperatura u otros requisitos del circuito. El más común de ellos es FR4 STD (*Flame Retardant 4 Standard*) con una constante dieléctrica aproximada de 4,6.

- **Prepreg:** El prepreg es la capa de material dieléctrico existente entre dos capas de cobre. Esta construido a partir de finas capas de entramado dieléctrico que le dan una estabilidad, grosor y constante dieléctrica determinada. Dependiendo del grosor de dicho entramado, la estabilidad en la constante dieléctrica será mayor o menor.

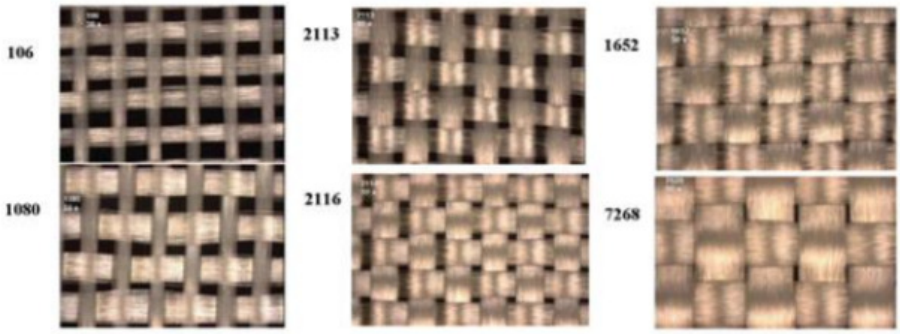



*Figura 19: Ejemplo de prepreg comerciales [54].*
*Existe una gran variedad y con diferentes grosores y constante dieléctrica. El ingeniero debe preguntar al fabricante la disponibilidad de estos materiales antes de empezar con el rutado de la placa. Ya que es necesario conocer el grosor entre plano de señal y plano de referencia. En la figura se puede ver los diferentes entramamos, el 7268 es el mas estable a nivel de impedancias*

Siempre que se vaya a diseñar una placa, es muy importante que antes de rutar la placa, se debe decidir qué empresa la fabricará. Existen empresas que serán muy económicas, pero puede que no tengan la capacidad suficiente para cumplir los requisitos mínimos del diseño.

Respecto al diseño de la placa modular del proyecto, y con las recomendaciones expuestas anteriormente se decidió que la PCB tendría las siguientes características:

- **Numero de capas**: Su número es muy importante en términos de coste, su mayor numero permite el rutado de placas más complejas y aislamiento de señales sensibles, por el contrario, el coste se incrementa en gran medida, en el caso del proyecto se han seleccionado 4 capas. Su distribución es la siguiente:

    - **Capas 1-4:** se utilizarán como capas de señal. En estas, irán rutadas todas las pistas de señales de alta velocidad, GPIO y comunicaciones además de las propias huellas de los componentes a soldar.

    - **Capa 2:** capa de cobre exclusiva para plano de masa, será la referencia de las pistas críticas del sistema.

    - **Capa 3:** se destinará principalmente a los planos de alimentación, con excepción de alguna pista residual que no pudo entrar en las capas destinadas a señal.

| Layer | Material Type | Thickness | |
|---|---|---|---|
| Top Layer1 | Copper | 0.035 mm | |
| Prepreg | 7628*1 | 0.2 mm | |
| Inner Layer2 | Copper | 0.0175 mm | 1.1mm (with copper core) |
| Core | Core | 1.065 mm | |
| Inner Layer3 | Copper | 0.0175 mm | |
| Prepreg | 7628*1 | 0.2 mm | |
| Bottom Layer4 | Copper | 0.035 mm | |

*Figura 20: Stackup de diseño modular para 4 capas y 1,6 mm de grosor, JLCPCB [55].*
*Este tipo de documentación es vital para un correcto ajuste de impedancias. Las placas multicapa se construyen a partir de la unión de capas de cobre mediante una capa de material dieléctrico, con un grosor especifico. Los emparedados resultantes se unen mediante núcleos de dieléctrico.*

- **Material:** FR4 con una constante dieléctrica de 4,6 aunque se tomará un valor inferior en el cálculo de impedancia debido a que la constante dieléctrica no es constante con la frecuencia, una aproximación es reducir en 0,2 con señales con una frecuencia aproximada 1 GHz en adelante.

- **Terminación de superficie:** HASL, es la más económica. Está destinada a prototipado y chequeo de un diseño, tampoco requiere la soldadura de componentes con encapsulado BGA por lo que no es necesario el uso de terminaciones más costosas como ENIG.

- **Distancia mínima entre cobre:** 0,2 mm exceptuando en la parte critica de la placa, donde se realizará un control de impedancia, en ese caso la separación será de 0,3mm.

- **Ancho de pista y vía:** Las pistas destinadas a señal serán de un grosor de 0,25 mm. Exceptuando aquellas que requieran un control de impedancia. Respecto a las pistas destinadas a alimentación el grosor será de 0,3 mm.
  Respecto al grosor de las vías, el taladro tendrá un diámetro de 0,26 mm mientras que la anchura total de la vía (incluyendo anillo y taladro) será de 0,6mm.

- **Prepreg:** En este caso se ha seleccionado el modelo 7628, este tipo de prepreg dispone de una mayor estabilidad en la constante dieléctrica a lo largo de su distribución por la placa, lo que impedirá perdidas en la integridad de señal en las partes críticas del circuito.

Como se ha comentado anteriormente, en todo el circuito se siguen las propiedades expuestas excepto en la zona más crítica del circuito, como es el bloque de comunicaciones relacionado con la recepción/transmisión de la señal RF del circuito y la disposición de la antena.

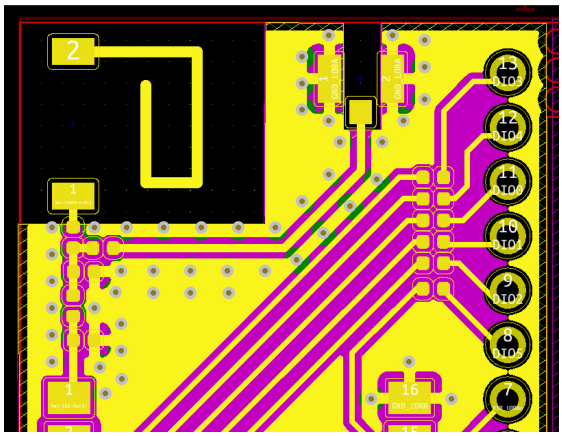


*Figura 21: Captura de distribución de antena cerámica y ULF para conectividad LoRa. Se trata de la parte más crítica de la placa en lo referente a control de impedancia. La integridad y ganancia total del transceptor depende en gran medida del ajuste realizado.*

El ajuste de impedancia sobre una antena en esta banda y según las instrucciones de fabricante, indican que la impedancia de la etapa de entrada debe ser lo más próxima a 50 ohmios para evitar en la medida de lo posible perdidas en la integridad de la señal. Existen muchas técnicas para este ajuste como puede ser el uso de una configuración *microstrip*, *stripline* o *coplanar wave*. Para este caso, se ha utilizado la configuración *coplanar wave* ya que se usa como referencia también el plano de masa que está en la misma capa que la señal, este plano de masa permite un aislamiento mayor de las capas internas del crcuito.

El control de impedancia en coplanar wave se basa en el ajuste de una pista con la modificación de 3 parámetros: El ancho de pista, la distancia del plano de la señal con el plano de referencia (capa 2 o plano GND) y la separación entre la pista y el plano de masa distribuido por la misma capa.

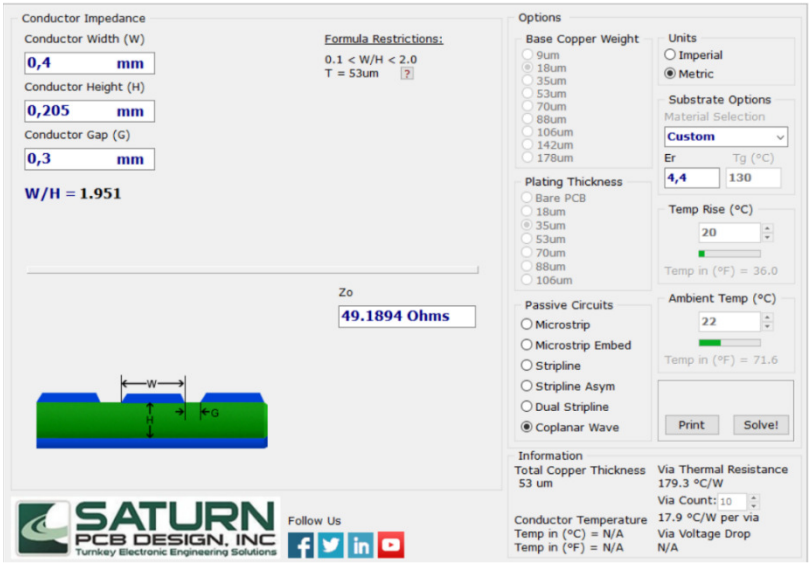



*Figura 22: Ajuste de impedancia para entrada de antena LoRa. Para estos cálculos, se ha utilizado la herramienta libre para fines no comerciales, Saturn PCB Design [56] Se puede observar los valores finales para esta parte del circuito. La impedancia teórica que obtenemos es 49,19 ohmios, un valor muy cercano al deseado.*

Tras el desarrollo del rutado de la placa, hay que chequear que se cumplen las reglas propuestas en el proyecto. A partir de ese punto se puede proceder con la generación de la documentación necesaria para la fabricación y montaje del diseño.
En la siguiente figura se puede observar el resultado final del rutado de la placa de circuito impreso.

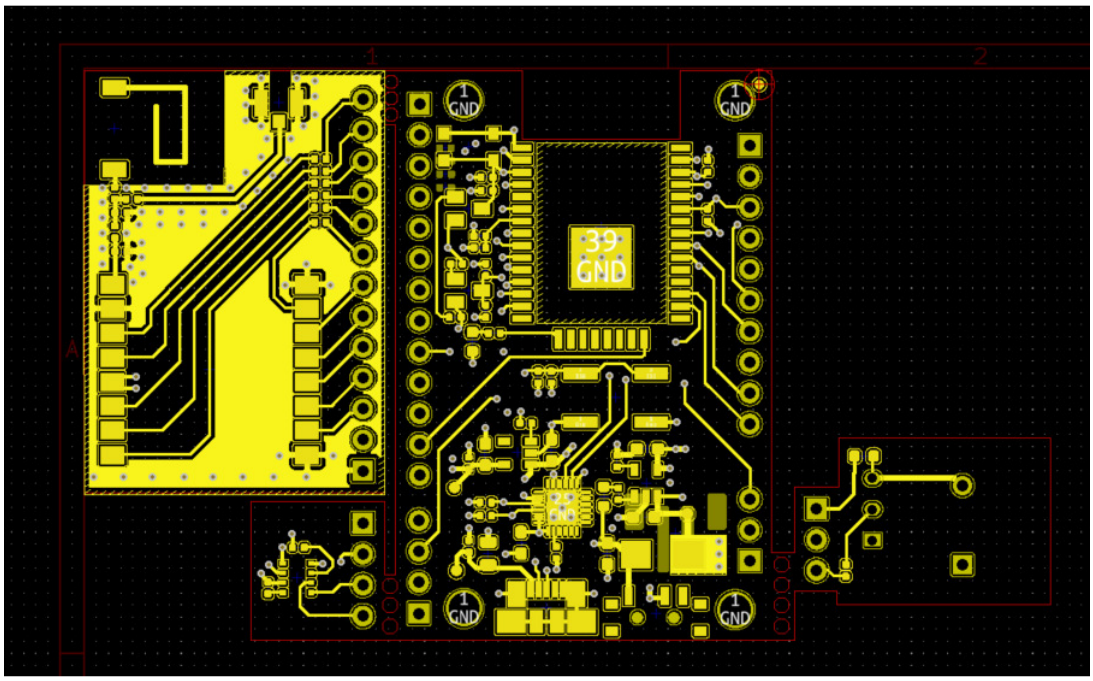



*Figura 23: Capa TOP de diseño PCB modular.*
*Se puede apreciar que a pesar de que tanto esquema como rutado del diseño se considera como una única placa de circuito impreso, esta se puede partir gracias a unos pequeños taladros para posteriormente utilizar aquellos módulos que convenga según la función que se le quiera dar. De esta manera se consigue un ahorro considerable en la fabricación y ensamblado de la electrónica*

## 4.4 Documentación/fabricación

Tras el desarrollo de esquemáticos, posterior diseño y rutado de la placa, es necesario generar la documentación esencial para el proceso de fabricación y montaje de los componentes de la placa. Es otro paso muy importante dentro del proceso de fabricación y ensamblado de una placa de circuito impreso ya que una buena documentación ahorra tiempo de producción y otros costes derivados.

### 4.4.1 Documentación para fabricación de PCB

Para la fabricación de la placa de circuito impreso es necesario que el diseñador aporte la siguiente información:

- **Ficheros Gerber:** Ficheros que contienen información de cada una de las capas que componen la placa. Aparte de las típicas capas que representa cada una de las capas de cobre, se pueden encontrar capas para el posicionamiento de la pasta de soldadura (para la fabricación de Stencil),

máscara, serigrafiado o contorno de la placa. También en algunos casos puede incluirse un fichero con notas informativas hacia el fabricante, como las zonas críticas donde poner especial cuidado en el ajuste de impedancias.

- **Ficheros de taladrado**: Se le podría considerar una capa más de información, donde se incluye el posicionamiento y diámetro de cada uno de los taladros a realizar en la placa, existe un fichero por cada tipo de taladro a realizar, pudiendo ser metalizado o no metalizado.

- **Fichero de Stack-up:** Se trata de un fichero en formato texto o grafico donde se indica la configuración de capas de cobre y dieléctricos de las que consta la placa de circuito impreso. Se debe indicar la cota de todas ellas, además de indicar cuales son las que se tienen que tomar como referencia para el ajuste de impedancia. Otros datos que se pueden incluir son los materiales dieléctricos, prepreg, dimensiones de pista, distancias entre elementos de cobre, aspecto ratio, etc.

### 4.4.2 Documentación para montaje de componentes en PCB

Teniendo disponible la placa fabricada, queda un solo paso más en el proceso de fabricación de un prototipo en lo referente a la electrónica, que es el montaje de los componentes en el sustrato. Si la placa es muy sencilla, está orientada a prototipado y no dispone ningún componente como los encapsulados BGA (que tienen una gran dificultad para soldar a mano) el propio ingeniero podría soldarlos para realizar pruebas. En caso de ser un diseño definitivo orientado a producción, hay que recurrir a empresas de ensamblaje que disponen de maquinaria de gran precisión para el montaje y soldadura de los componentes.
Para poder recurrir a este tipo de servicios, es necesario aportar la siguiente información:

- **Ficheros de posicionamiento o *pick and place***: Se trata de ficheros con toda la información referente al posicionamiento de cada uno de los componentes de la placa. No solo incluye la posición en los dos ejes de coordenadas, también incluye información sobre si el componente está en la capa TOP o BOTTOM de la placa o en ángulo de posición.

- **Plano de montaje**: Es un documento donde se puede observar las capas de montaje con la referencia asociada a cada componente. También se incluyen marcas que ayudan al montador a identificar el pin número 1 de cada integrado, para soldar en la polaridad correcta cada componente.
  Dentro del *Anexo A: Documentación relativa a diseño de modulo/sensor se puede encontrar el plano de este diseño.*

- **Fichero BOM** (*Bill Of Materials*) **o lista de materiales:** Se trata de un documento, normalmente en formato Excel, que incluye toda la información relativa a cada integrado a soldar en la placa. Aunque en una placa se haya diseñado para albergar una cantidad determinada de componentes, no siempre tienen porque estar soldados, un ejemplo podría ser las resistencias de configuración de ciertos componentes. En ese caso, es la lista de materiales la que indica que referencias deben ser

las que se suelden en la placa y que integrado, fabricante y encapsulado se debe soldar en cada huella. También suele incluirse una pequeña descripción de cada componente, así como un enlace a su hoja de características o distribuidor de este.
*Anexo A: Documentación relativa a diseño de modulo/sensor se puede encontrar el plano de este diseño.*

### 4.4.3 Otros documentos

Otro documento importante en el proceso de fabricación de un producto electrónico y que no siempre es tenido en cuenta, es el fichero CAD (*Computer-Aided Design*) de la placa electrónica. Este tipo de ficheros son de gran ayuda para el proceso de diseño y fabricación de la carcasa del producto. El ingeniero mecánico encargado del desarrollo mecánico tendrá una representación virtual de cómo será el producto electrónico montado, de tal modo, que podrá realizar todas las comprobaciones pertinentes antes de la fabricación de la mecánica.

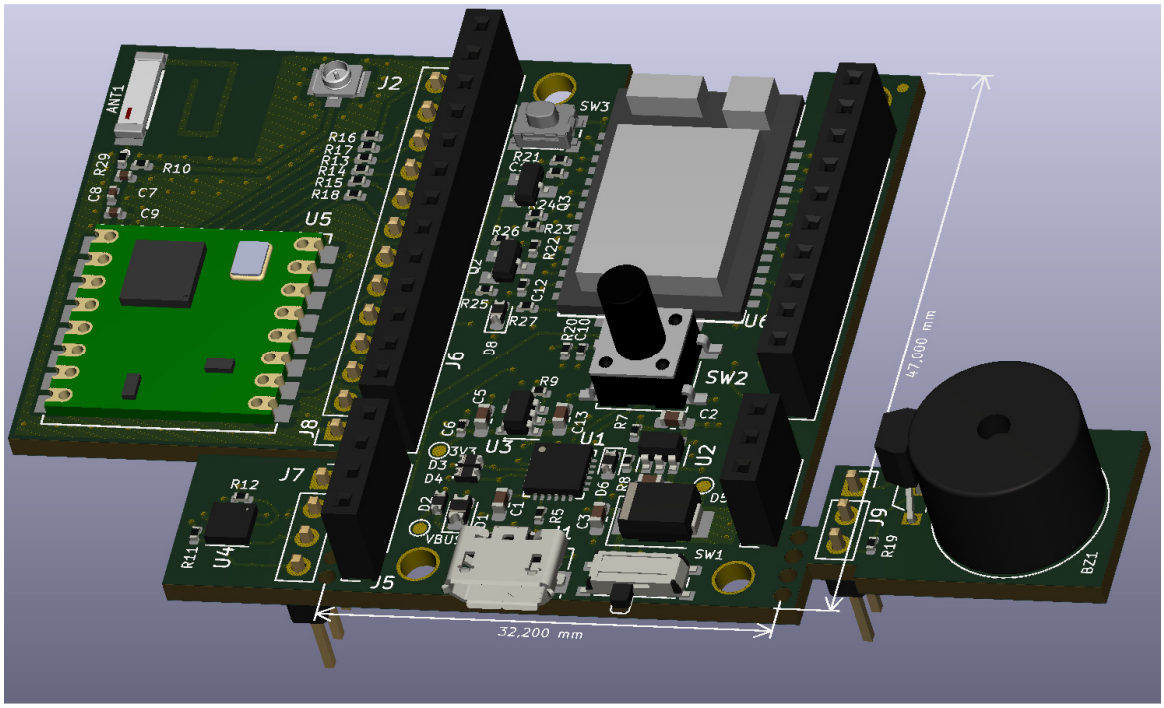



*Figura 24: Representación 3D de PCB modular de nodo/sensor.*
*Es de especial utilidad para interacción del ingeniero mecánico a la hora de diseñar una mecánica.*

### 4.5 Montaje y puesta en marcha

Tras la recepción de componentes como de la placa de circuito impreso, es necesario el montaje de todos ellos. Al tratarse de un prototipo se realizará de forma manual.

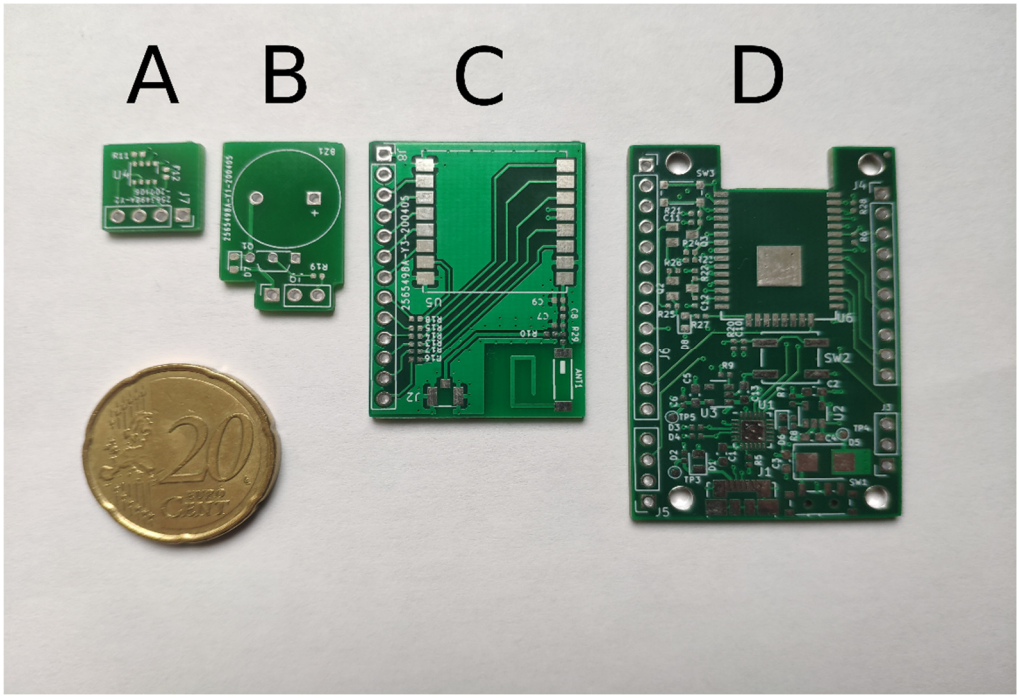



*Figura 25: PCB diseñada en este trabajo.*

En la figura 25 se pueden observar las placas diseñadas para cubrir las necesidades de nodo/sensor y gateway LoRa. Las placas A, B y C son los módulos periféricos insertables con las funciones de sensor, Buzzer y transceptor LoRa respectivamente. La placa D se trata de la placa principal del sistema, con la función de unidad de procesamiento, comunicación 2,4GHz y alimentación.

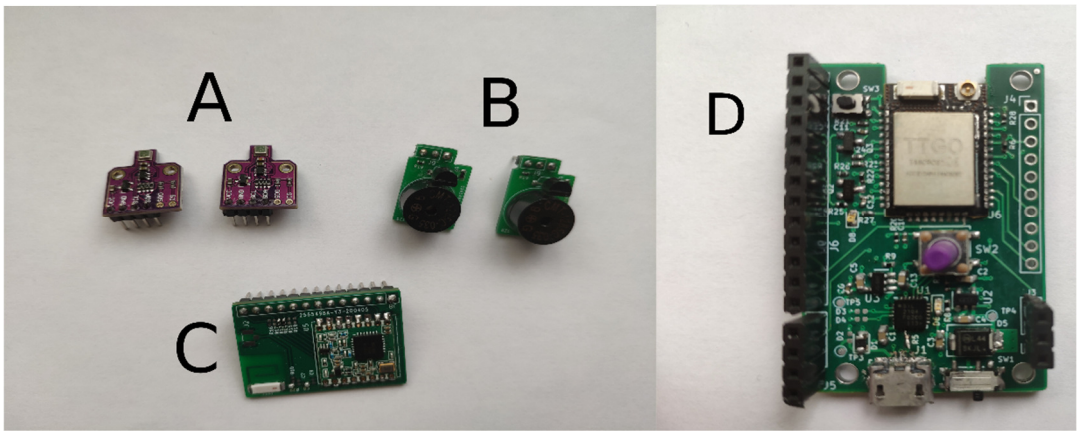



*Figura 26: PCB principal y módulos accesorios montados y verificados eléctricamente.*

En la figura 26, a la derecha se encuentra el módulo D, se trata de la placa principal del prototipo, los módulos A, B, C son los modulos accesorios montados y listos para su conexión con la placa principal.

- **Configuración 1**: Buzzer(B)+ Sensor(A) + Principal(D) = Nodo/Sensor
- **Configuración 2:** Modulo LoRa(C)+ Principal(D) = Gateway LoRa

En el proceso que se realiza un montaje de forma manual, es recomendable ir comprobando con el polímetro la conductividad entre cada una de las alimentaciones más importantes del sistema con respecto a la masa del circuito, de esta manera, se puede detectar cualquier fallo de soldadura rápidamente sobre la marcha.

Tras el montaje de todos los componentes y periféricos, ya solo resta conectar la placa principal a un cable de alimentación USB y comprobar que todas las alimentaciones de la placa dan unos valores correctos, estos son VBUS, 3V3 y VBAT. Tras su verificación, se puede proceder con la conexión de la placa con un ordenador. Inmediatamente después de su conexión, el equipo debe reconocer la conexión de un nuevo dispositivo y aparecer este en el listado de administración de dispositivos del panel de control de Windows como un nuevo puerto COM. Este puerto permitirá al usuario realizar las descargas de firmware del dispositivo, así como realizar tareas de depuración.

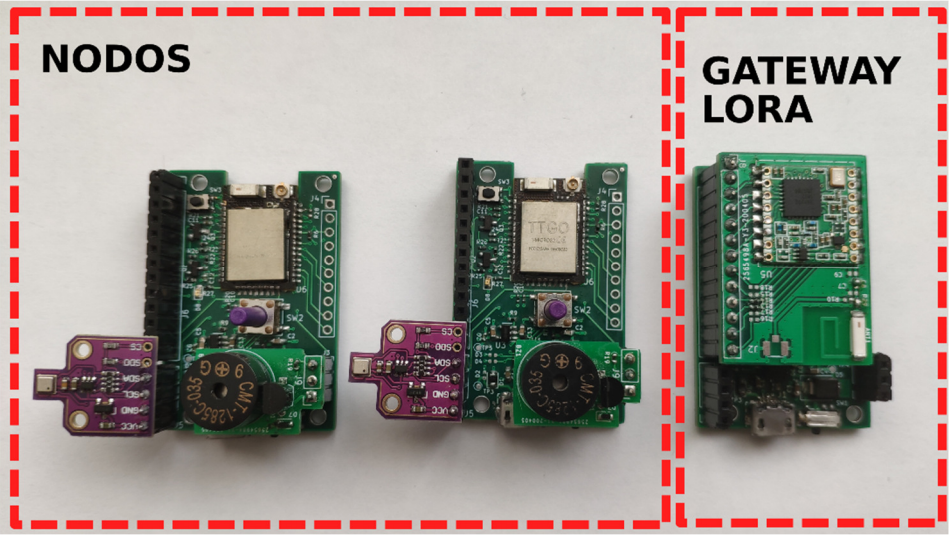



*Figura 27: Dispositivos preparados para su puesta en marcha y posterior integración con mecánica y batería.*

Tras realizar una primera descarga al sistema, la posibilidad de tener un fallo importante a causa del proceso de montaje es bajo, por lo que solo queda realizar el montaje de los módulos periféricos en los zócalos laterales de la placa principal y realizar las pruebas de funcionamiento.

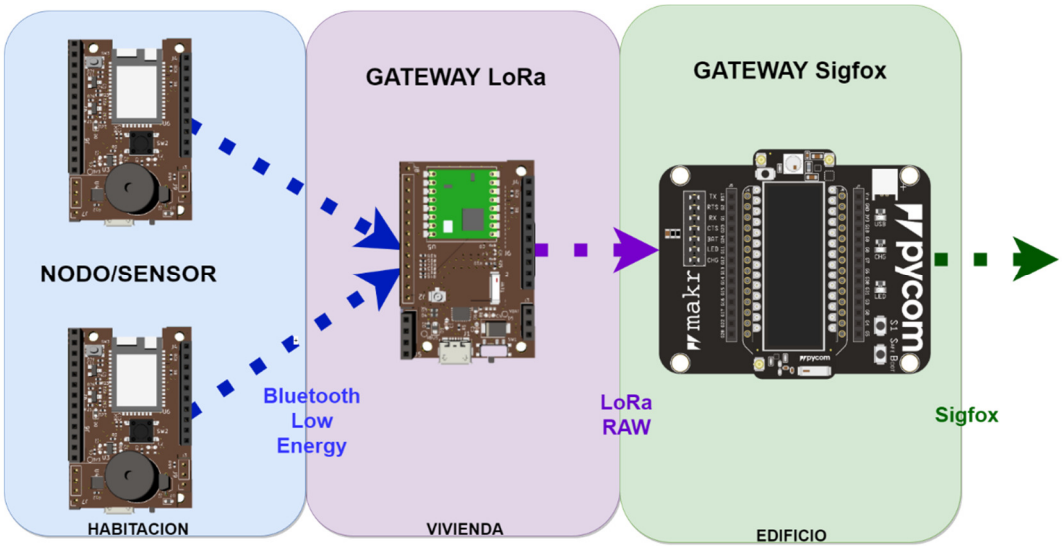



*Figura 28: Diagrama de dispositivos de la arquitectura.*

En la figura 28 se puede apreciar el sistema completo con todos los dispositivos que intervienen en él. La comunicación entre ellos se realiza de forma inalámbrica a través de bluetooth low energy, LoRa RAW para finalmente comunicar a la nube a través de Sigfox.

### 4.6 Programación e incorporación de los prototipos al sistema.

La programación de estos dispositivos respecto a la realizada en los prototipos basados en soluciones comerciales no difiere y por lo tanto se pueden utilizar las mismas herramientas y proyectos. Las únicas modificaciones que hay que realizar cuando se programe alguna de estas placas es quitar la etiqueta PCB_COMERCIAL al principio de cada código, para que al compilar la asignación de los pines sea la apropiada para la placa.

*Todo el código del firmware para la programación de estos dispositivos está disponible en el ANEXO B del documento.*

# 5. Desarrollo de sistema de emergencias

A pesar de haber realizado un diseño propio para el desarrollo de la arquitectura propuesta en este trabajo, debido al tiempo limitado que se dispone para el desarrollo de este, es necesario paralelizar en la medida de lo posible la mayor cantidad de tareas, ya que el proceso de fabricación y montaje de una placa de circuito impreso implica una espera hasta la recepción del producto. Además, antes de cualquier proceso de producción es necesaria una etapa de prototipado y prueba de concepto de la idea. Por lo tanto, el desarrollo de firmware y la primera prueba de concepto del sistema se realizará con placas comerciales similares a la diseñada con modificaciones sobre las mismas para cumplir con los aspectos básicos de los dispositivos del sistema.

## 5.1 Introducción a la arquitectura del sistema emergencias

La arquitectura del sistema de emergencias se puede dividir en 3 capas a nivel de hardware y comunicaciones. Cada una de ellas esta comunicada con la siguiente mediante una tecnología de comunicación WPAN o LPWAN diferente. La primera capa, está formada por un conjunto de nodos/sensores con conectividad bluetooth. Estos sistemas se encargarán de tomar parámetros del entorno o de un pulsador de emergencia disponible en cada dispositivo. El principal parámetro será la temperatura, aunque también tomará medidas de calidad del aire asociados a la concentración de diferentes gases que son perjudiciales para la salud. Si la temperatura en el area a controlar supera un umbral limite (unos 50-60ºC ), el nodo enviará un mensaje a través de la conexión establecida con la siguiente capa, un dispositivo que actuará como puerta de enlace y que funcionará como cliente bluetooth a la vez que da acceso a los elementos de la subred WPAN a una red LPWAN, en este caso LoRa.

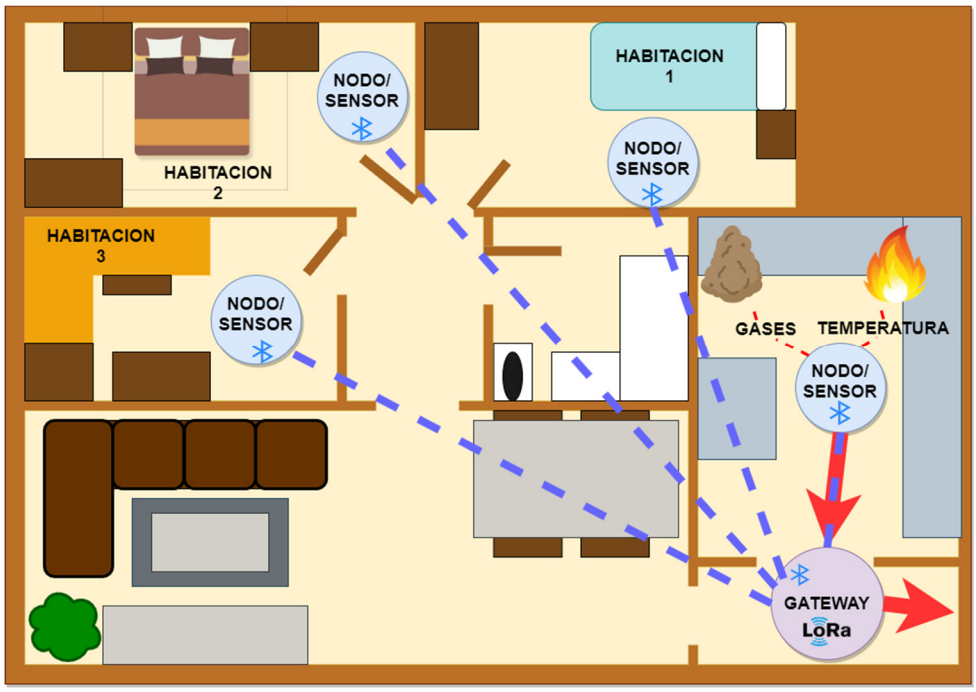



*Figura 29: Capa 1. Red WPAN para vivienda. plano de una vivienda.*

*Cada uno de los sensores instalados controla los habitáculos más importantes de la vivienda. Todos ellos se encuentran conectados al Gateway de la vivienda para comunicar solamente en el caso de activación de dicha alarma. En el caso de la imagen, al detectar un incendio en la cocina, el nodo responsable envía la información a través del Gateway asignado.*

La segunda capa está formada por la red de dispositivos con conectividad LoRa en una topología en estrella. Cada vivienda dispone de un Gateway que actuará como cliente BLE recibiendo la información procedente de cada servidor de la subred WPAN que existe. Todos estos dispositivos estarán conectados a su vez con otra puerta de enlace que actuará como cliente LoRa dando acceso al sistema a la nube a través de Sigfox. La necesidad del uso de LoRa en esta capa es por su gran alcance a pesar de los obstáculos existentes en un edificio. El uso de una tecnología WPAN sería inviable debido a la dramática reducción de cobertura en presencia de obstáculos como las paredes o pisos del edificio.

*Figura 30: Representación de Capa 2 y 3 de arquitectura.*
*Comunicación entre ambas mediante LPWAN en topología estrella. En la vivienda 2 se detecta un incendio y el nodo envía un mensaje de alarma a través de su puerta de enlace asignada. Esta puerta de enlace actúa como servidor LoRa enviando el mensaje al dispositivo de la azotea que comunicara con la nube.*

Finalmente, tras el envío del mensaje por parte del dispositivo con conectividad Sigfox que está asignado al edificio, este viajará a través de la red estaciones distribuidas por la compañía propietaria hasta su nube, donde quedará registrada toda la información enviada correlada con la ID del dispositivo registrado. A partir de este punto existen varias opciones para gestionar esos datos. De forma genérica esos datos se extraen y se envían a un servidor gestionado por el cliente y almacenados en una base de datos para el posterior cruce de datos y obtener más información sobre la vivienda que ha disparado la alarma. Por último, la aplicación de visualización aportará al usuario del centro de emergencias la información pertinente para su respuesta o intervención.

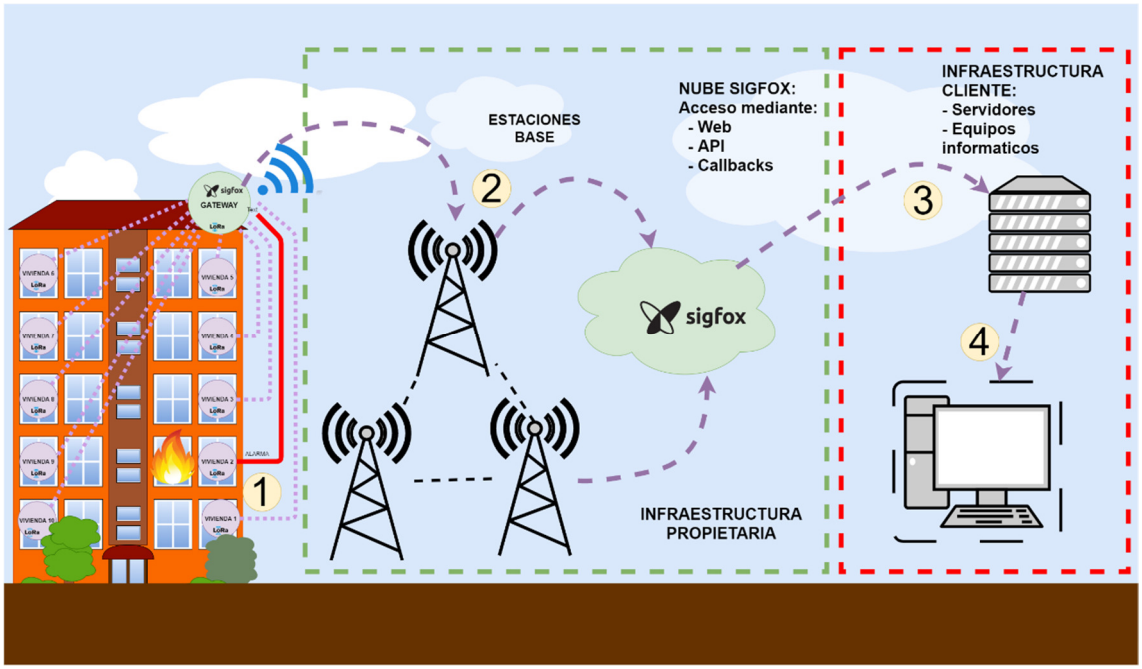



*Figura 31: Visión global de sistema de emergencias.*
*El uso de una infraestructura propietaria como Sigfox para su transmisión y almacenamiento, reduce en gran medida los costes iniciales de implantación y mantenimiento del sistema.*

En la figura 31 se puede ver un diagrama global de todas las partes que intervienen en el sistema de emergencias. Se podría clasificar en 4 estados secuenciales:

1. Proceso de comunicación de la alarma detectada en la vivienda al gateway Sigfox situado en la azotea.
2. Envío de la trama Sigfox a través de la red de estaciones propietarias de la compañía Sigfox hasta sus servidores dedicados.
3. El servidor de Sigfox siguiendo las ordenes de Callback envía la información al servidor del cliente.
4. El usuario del centro de emergencias recibe la información del servidor con el cruce entre BBDD realizado, obteniendo la dirección exacta donde se ha producido la emergencia.

## 5.2 Elementos del sistema

Para el desarrollo del sistema necesitaremos buscar alguna alternativa comercial que permita realizar la prueba de concepto. Para ello se inicia la búsqueda de varias placas genéricas en Aliexpress [57] basadas en el mismo bloque de procesamiento que el propuesto en el diseño, el microcontrolador ESP32.

Para cumplir con toda la estructura del sistema de emergencias se necesitará:

1. **Prototipo sensor/nodo:** Placa genérica ESP32 con conectividad Bluetooth 4.2 + Gestión de batería + Placa con sensor de temperatura. Para el prototipado de este elemento se utilizará la placa Lilygo TTgo T5 V2.4 [58] con display de tinta electrónica. Dispone de un ESP32 pico D4, como el utilizado en la placa modular diseñada. Dispone de gran cantidad de GPIO accesibles, así como altavoz y gestión de batería. Para conseguir simular su funcionamiento en un lugar sin alimentación y de forma autónoma se conectará una batería LiPo de 3,7V 650 mA para drones de vuelo y para la recepción de parámetros, se usará una placa genérica con el sensor BME680.

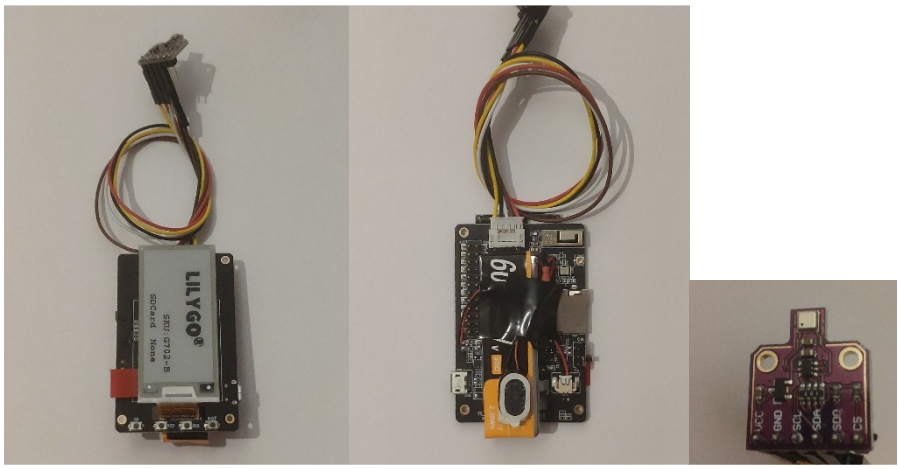



*Figura 32: Prototipo de nodo/sensor a partir de placa genérica + BME680+ altavoz + Batería*

2. **Para Gateway LoRa:** Placa genérica ESP32 con conectividad bluetooth 4.2 + transceptor LoRa + Gestión de batería:
   En este caso se utilizará la placa TTgo ESP32 LoRa 2.4V de Lilygo [59] la cual dispone de transceptor LoRa además de display OLED, antena y gestión de batería. El procesador utilizado en este caso es también el ESP32 por lo que ya de forma nativa también incluye conectividad bluetooth 4.2.

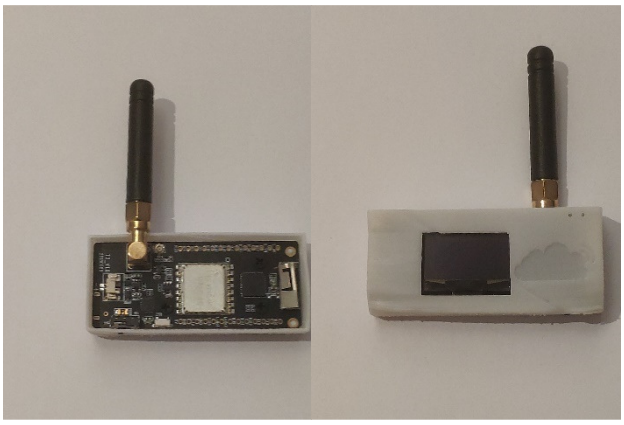


*Figura 33: Prototipo gateway LoRa a partir de placa TTgo T3 + Batería*

3. **Gateway Sigfox:** Placa con transceptor LoRa + Transceptor Sigfox + Gestión de batería.
   Para este caso, se ha recurrido a una de las pocas opciones disponibles con conectividad Sigfox que existen de forma fácilmente accesible en el mercado, el módulo lopy4 **[60]** que estará alojado sobre la placa de expansión para gestión de batería Pymakr v3 **[61]**. Al igual que los casos anteriores su núcleo está basado en un ESP32 con además conectividad LoRa y Sigfox. También, dispone de librerías propias en Python lo que facilita aún más su programación y puesta en marcha. Este sistema deberá estar cableado o disponer de un sistema de alimentación por batería con gran capacidad además de estar apoyado por algún sistema de *energy harvesting* (un ejemplo sería un panel fotovoltaico).

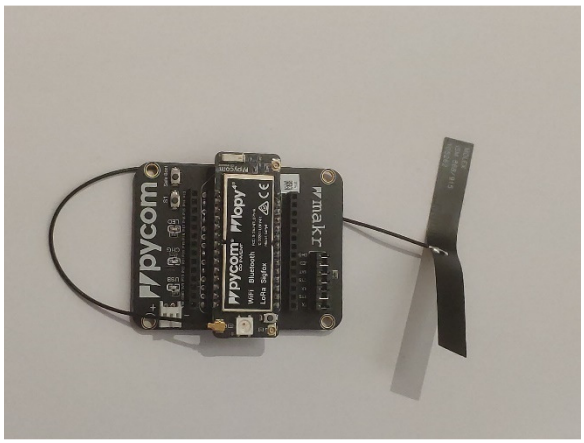



*Figura 34: Gateway Sigfox. Lopy4 de Pycom + Placa de expansión + Batería*

## 5.3 Implementación de arquitectura.

A continuación, se expondrá el desarrollo de la arquitectura del sistema. Esta se ha desglosado por 5 fases bien diferenciadas, como se indicó en el apartado 1.2 del documento, principalmente debido a su complejidad.

### 5.3.1 Fase 1: Lectura de parámetros de medida en sensores y generadores de alarma en el nodo.

El objetivo de la primera fase se basa en la programación del nodo de tal manera que cada cierto periodo de tiempo realice la recolección de datos propuestos y haga un análisis sobre ellos. Si los datos son correctos, el sistema entrará en un estado de ahorro de energía (en este caso Deep sleep), hasta la siguiente lectura programada, donde el dispositivo arrancará desde el principio. Este periodo de tiempo es totalmente configurable y estará fijado en un valor del orden de 1 minuto.

Por otro lado, cada nodo funciona como un servidor dentro de una red WPAN con tecnología BLE en topología en estrella. Cuando el dispositivo no se encuentra conectado, se anunciará a la espera de establecer una conexión con el cliente del sistema. Ya realizada la conexión, el sistema tomará medidas y realizará notificaciones al cliente cuando alguna de las medidas supere el umbral estimado.

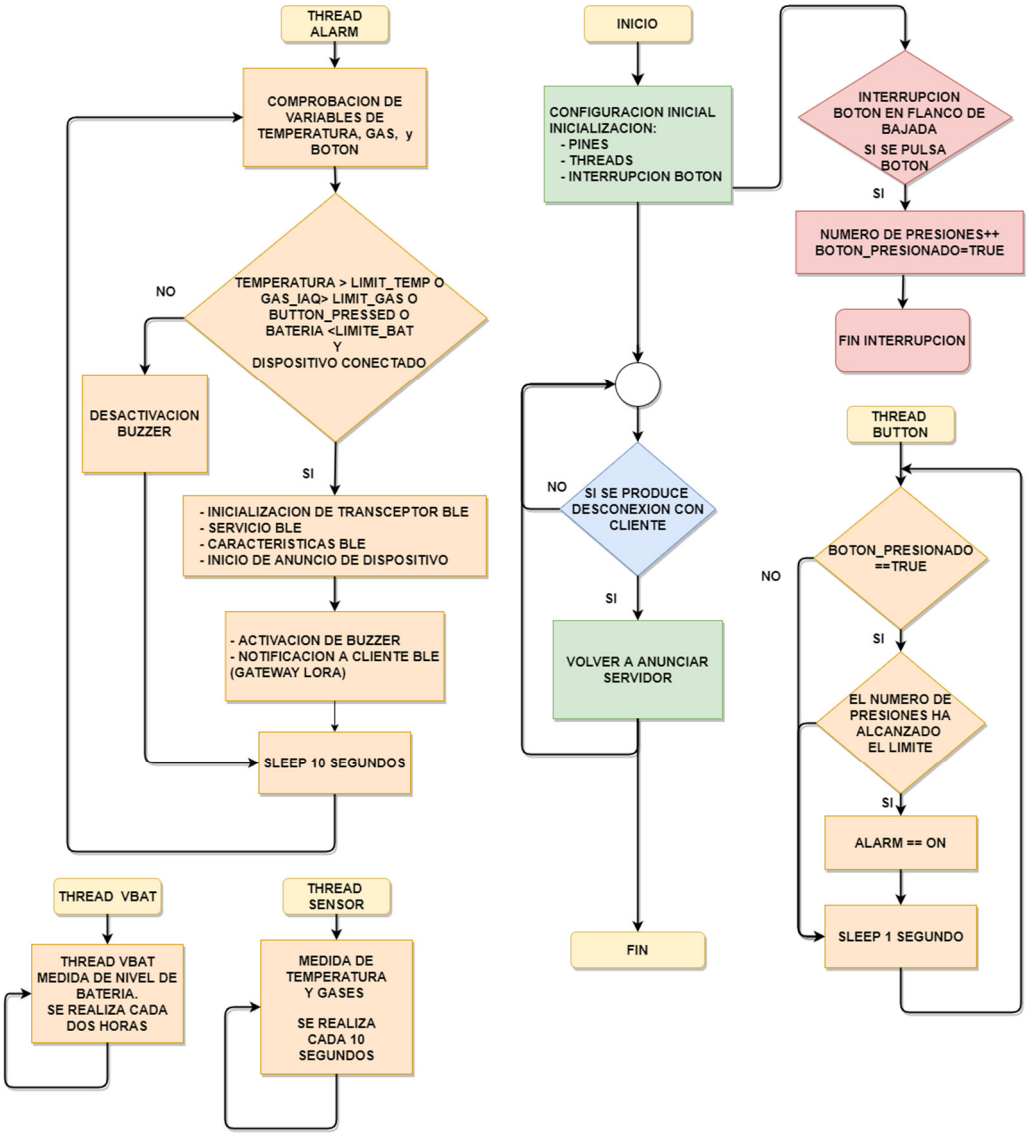



*Figura 35: Diagrama de flujos de firmware nodo/sensor*

Como se puede observar en el diagrama de flujos de la figura 35, el firmware se basa en un hilo principal en el que se realiza la configuración de todos elementos integrantes del sistema, como pines, PWM (*Pulse-Width Modulation*), etc. Además, ejecutará 4 hilos diferentes para realizar las tareas de control de forma independiente y simultánea.

Estos hilos controlarán:

1. **Nivel de batería:** A través de un ADC del microcontrolador, el dispositivo puede saber el valor en voltios que hay en la batería. Si el valor de la batería se encuentra por debajo de 3,7V es necesario su recarga de inmediato.

2. **Control de parámetros medioambientales:** El sensor que dispone el nodo toma medidas de temperatura y gases mediante el parámetro denominado IAQ.

3. **Control de pulsaciones en el botón de emergencia**: Pulsando de forma repetitiva el pulsador en un determinado periodo de tiempo, también activa una alarma de emergencia. Esta alarma está controlada por un hilo y una interrupción, que realizan el control temporal y de pulsación respectivamente.

4. **Hilo de control de alarmas:** realiza un chequeo de cada una de las fuentes de activación de alarma y de superar alguno de los límites establecidos, lo notificará al gateway por BLE.

Respecto al consumo de este dispositivo, debe ser el más bajo posible. Es el dispositivo más crítico del sistema en este punto, ya que debe ser un elemento totalmente autónomo con la posibilidad de emplazamiento en zonas de difícil acceso, por lo que su autonomía debe estar garantizada durante un largo periodo de tiempo.

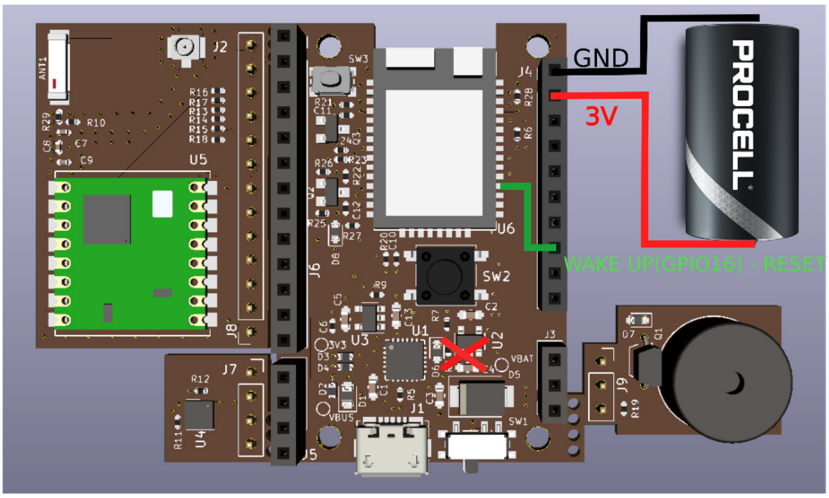



*Figura 36: Modificaciones en placa para trabajar en modo Deep sleep.*
*Realizando los cambios propuestos en la imagen, se consigue que el equipo consuma 10uA la mayor parte del tiempo. Solamente en el proceso de equipo despierto en el que se realizará la lectura de parámetros tendrá un consumo elevado.*

Para conseguir el funcionamiento del modo Deep sleep y un mayor ahorro en modo despierto, se deben adoptar las siguientes medidas:

1. **Reducir la frecuencia de trabajo del sistema.** La frecuencia de trabajo máxima del microcontrolador es de 240 MHz. Pero en este caso no es necesaria tanta velocidad de computación, con 80 MHz el dispositivo funciona correctamente y permite ahorrar energía.

2. **Configurar el dispositivo para trabajar en modo Deep sleep.** Es necesaria la modificación de la placa, para que el equipo pueda reiniciarse cada vez que despierte debido al timer. Esta modificación se trata de conectar el pin de wake up (GPIO16) del circuito a la entrada de reset de tal manera que, cumplido el tiempo de suspensión, el equipo pueda reiniciarse.

3. **Inhabilitar el bloque de alimentación por defecto del sistema.** y conectar una pila de litio de 3V para evitar la pérdida de energía por los integrados no esenciales de la placa.

En la siguiente tabla se puede observar el consumo estimado del dispositivo:

| Casos y Batería | Periodo medida (s) | Consumo (mAh) | Tiempo de autonomía (días) |
|---|---|---|---|
| **Caso 1: Batería 3,7V 650mA (T + Gas)** | 30 | 2,169 | 12,48655294 |
| | 60 | 1,136 | 23,84096244 |
| | 120 | 0,624 | 43,40277778 |
| **Caso 2: Batería 3,7V 650mA (T)** | 30 | 1,768 | 15,31862745 |
| | 60 | 0,939 | 28,8427405 |
| | 120 | 0,524 | 51,66603078 |
| **Caso 3: Pila Litio 3V 1480mA (T + Gas)** | 30 | 2,060 | 29,93527508 |
| | 60 | 1,030 | 59,87055016 |
| | 120 | 0,516 | 119,5090439 |
| **Caso 4: Pila litio 3V 1480mA (T)** | 30 | 1,660 | 37,14859438 |
| | 60 | 0,830 | 74,29718876 |
| | 120 | 0,416 | 148,2371795 |

*Tabla 7: Comparativa de consumo y autonomía de nodo según parámetros de medida y fuente de alimentación.*

Se puede apreciar en la Tabla 7, se analizan 4 casos diferentes. En ninguno de ellos se ha tenido en cuenta el consumo producido cuando se genera una alarma, ya que este evento se considera como un acontecimiento extraordinario y, por lo tanto, hay muy pocas probabilidades de que ocurra, lo que no influiría en el consumo general del sistema (En caso de alarma, el consumo se dispara a causa de la comunicación bluetooth y la activación de la alarma sonora, por lo que se será necesaria su recarga). Tampoco se considera el consumo producido cuando se transmite la alarma por bluetooth ya que el transceptor esta deshabilitado hasta producirse el evento de emergencia.

En los casos 1 y 3 se ha tenido en cuenta la medida de dos parámetros, temperatura y calidad del aire, en cambio en los casos 2 y 4 solamente se toman medidas de temperatura, que es el parámetro principal de detección.
Los casos 1 y 2 se ha tenido en cuenta el consumo de la parte de regulación de tensión que se produce para adaptarla a un valor valido del circuito, el cual se estima en unos 100 uA y asumiendo que ningún otro elemento introduce consumo adicional en el circuito.
En cambio, los casos 3 y 4 al estar alimentado por una pila de litio a un valor de tensión apto para el funcionamiento del circuito, no es necesaria la unidad de alimentación y regulación y por lo tanto no se produce ese consumo adicional, consiguiendo un 12,5 % más de autonomía. Aun así, una autonomía de 148 días es un valor muy bajo para este tipo de sistemas, siendo el principal responsable y factor determinante de este exceso de consumo la unidad de procesamiento, cuyo consumo en modo activo es de unos 25 mA.

### 5.3.2 Fase 2: Comunicación WPAN BLE nodos-gateway.

En la siguiente frase se pretende desarrollar la comunicación de los diferentes dispositivos nodos con el Gateway LoRa para la formación de una red WPAN, en este caso se utilizará bluetooth low energy.

Antes de explicar todo el proceso de comunicación, así como el diagrama de flujos del Gateway LoRa es necesario realizar una introducción de los elementos más importantes de esta tecnología para poder así entender y explicar el papel de cada dispositivo dentro de la red, así como también los métodos de transferencia de datos y los identificadores de característica y método.

**GAP** *(Generic Access Profile):*
Se trata de un perfil que se encarga de controlar las conexiones y anuncios en bluetooth low energy. Este tipo de perfil permite configurar las propiedades del dispositivo y determina como los dispositivos dentro de una red pueden interactuar.

Existen dos clasificaciones de los dispositivos según sus roles y funcionalidades en la red:

**Según el rol de conexión**, existen dos papeles principales dentro de su rol en una red BLE:

- **Máster/Central:** el dispositivo BLE será el que realiza la búsqueda de otros dispositivos miembros de la red e inicia la conexión con el dispositivo anunciante.

- **Esclavo/Periférico:** Es el dispositivo que dentro de la red acepta la solicitud de conexión. Normalmente empiezan anunciándose y esperando una solicitud de conexión.

**Según su rol en funcionalidad GATT** *(General Attribute Profile)* la cual define de qué modo dos dispositivos BLE pueden comunicarse utilizando los servicios y características. Esta comunicación se realiza mediante el protocolo ATT que se usa para almacenar dichos servicios y características.

Existen dos roles según la funcionalidad GATT:

- **Cliente:** Es el dispositivo que accede a los datos en el servidor a través de operaciones de lectura, escritura, notificación o indicación

- **Servidor:** Es el dispositivo que almacena los datos localmente y proporciona los métodos de acceso a los datos a un cliente GATT.

Un concepto de gran importancia es la relación entre servidor y cliente, El cliente envía solicitudes al servidor. Todas las transacciones son iniciadas por el dispositivo maestro/cliente que recibirá la respuesta del esclavo/servidor. Al establecerse la conexión, el periférico sugiere un intervalo de conexión al dispositivo central para solicitud de nuevos datos disponibles.

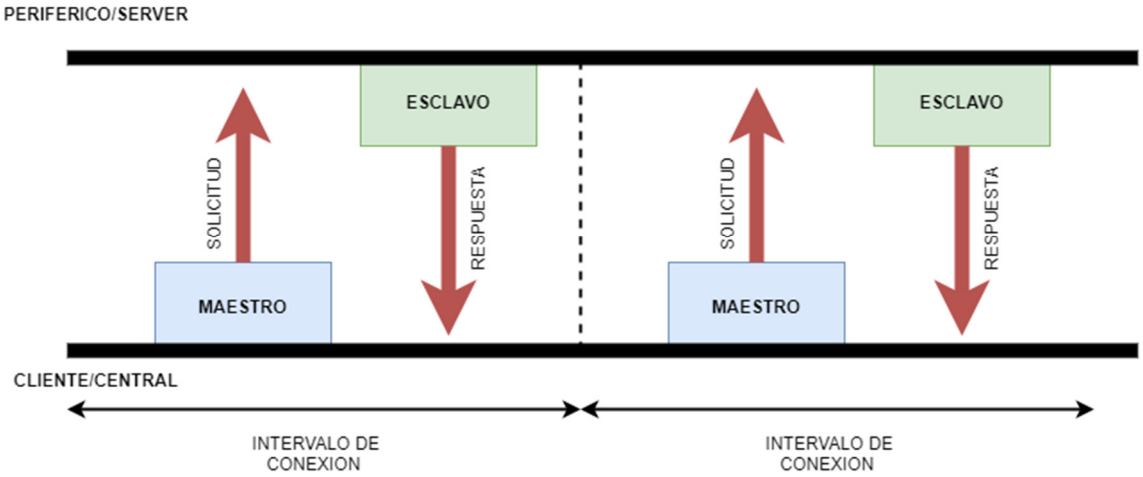



*Figura 37: Mecanismos de interacción en la comunicación entre dispositivos dentro de una red BLE.*

**Perfil, Servicios y características.**
Las comunicaciones en BLE están basadas en objetos de alto nivel denominados perfiles, servicios y características.

Los **perfiles** son colecciones predefinidas [62] de servicios creadas por el Bluetooth SIG o fabricante del periférico. Por ejemplo, el perfil de frecuencia cardiaca combina el servicio de frecuencia cardiaca con el servicio de información de dispositivos.

Los **servicios** contienen fragmentos de datos denominados características. Cada servicio puede contener una o varias características y pueden ser identificados mediante un valor numérico único denominado UUID (*Universally unique identifier*), este código puede tener dos tamaños 16 bits para los servicios adoptados oficialmente [63] o 128 bits para servicios personalizados.

Las **características** se tratan del nivel más bajo dentro de las transacciones GATT, son aquellas que encapsulan un tipo de dato. De igual manera que los servicios, las características se pueden distinguir mediante un identificador único con un formato que depende de si ha sido generado oficialmente [64] o es un

identificador personalizado. Se trata del principal objeto para interaccionar con el periférico BLE, y pueden ser de lectura o escritura.

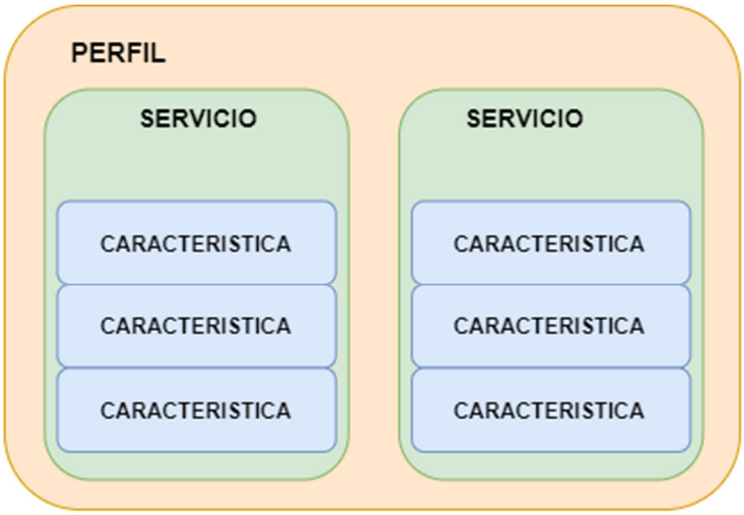



*Figura 38: Representación de un perfil BLE.*
*Un perfil puede contener varios servicios y estos a su vez varias características.*

Dentro del sistema propuesto en el proyecto, la arquitectura de la red WPAN que forma el primer nivel, el dispositivo nodo es el que recogerá los parámetros de temperatura y gas, y, por tanto, será el dispositivo que actuará bajo el rol de periférico/esclavo con una funcionalidad de servidor, es decir, será el dispositivo que se anuncie en la red y esperará a la solicitud de conexión realizada por el máster. Cuando detecte alguna alarma, almacenará dichos datos y proporcionará los métodos de acceso para el cliente.

En referencia al gateway LoRa, su papel principal dentro de la red es la de actuar como rol de máster/central y funcionalidad de cliente. Reiterativamente se encargará de escanear los dispositivos anunciados dentro de su cobertura y realizará solicitudes de conexión a los dispositivos siempre y cuando el identificador de servicio de estos coincida con el de identificador de servicio asignado a los dispositivos de detección de emergencias, de esta manera, solamente los dispositivos del sistema de emergencia podrán interactuar con el cliente. Por lo tanto, cada red WPAN existente en cada vivienda deberá tener su propio identificador de servicio para evitar la conexión a dispositivos cercanos situados en otra vivienda. Cuando reciba algún dato de alarma proveniente de alguno de los nodos pertenecientes a la red, el Gateway enviará una trama a través de LoRa RAW para avisar a la última capa de comunicación del sistema.

Respecto a otras características del gateway cabe destacar que al tratarse de un dispositivo que está constantemente escaneando dispositivos del servicio de emergencia, no es posible una gran autonomía debido a su elevado consumo y por lo tanto deberá ser un dispositivo cableado. Aun así, es recomendable la incorporación de un sistema de alimentación complementario basado en batería de litio para dotarle de la autonomía suficiente ante posibles cortes en la red eléctrica.

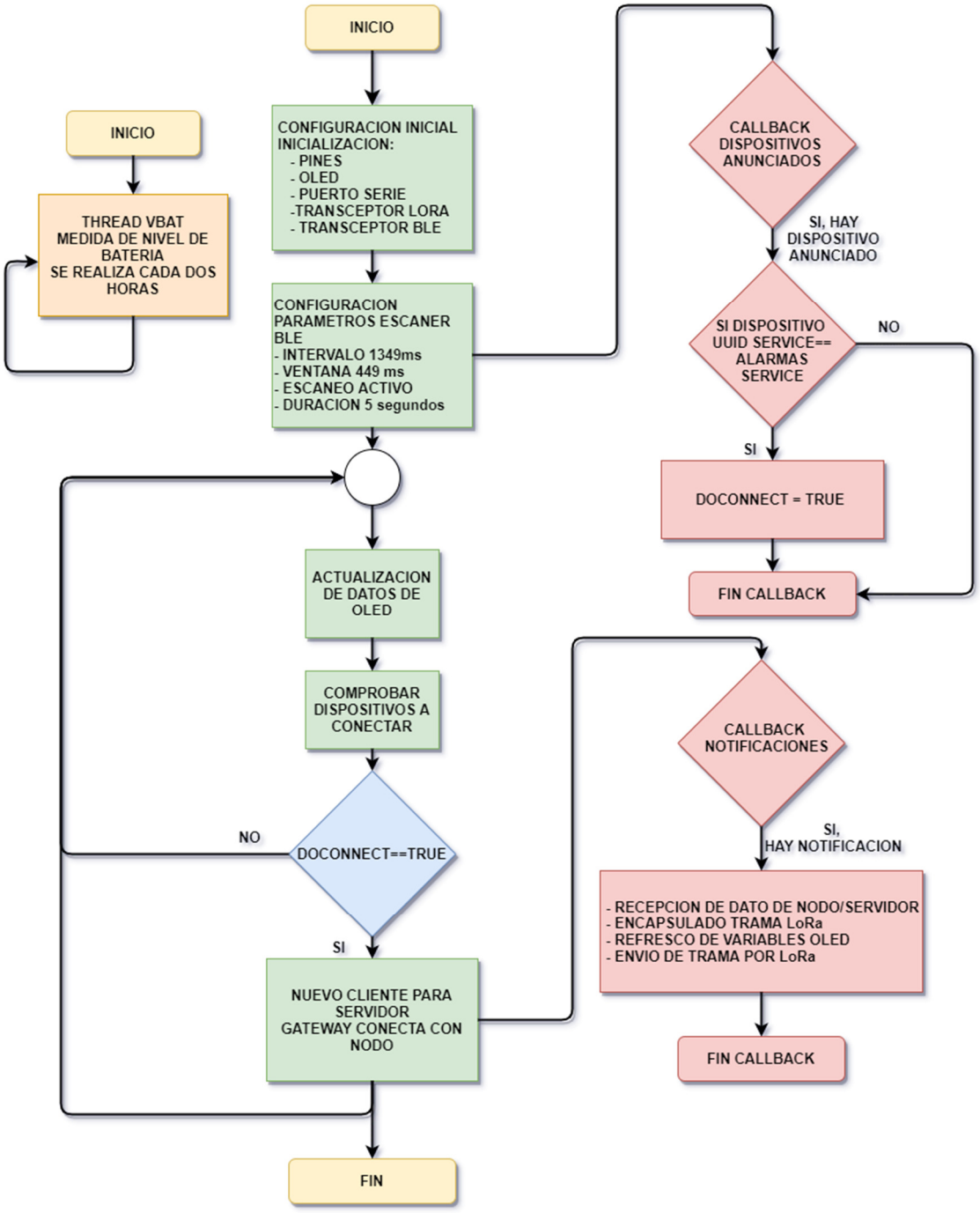



*Figura 39: Diagrama de flujos de firmware de gateway LoRa*
*El dispositivo funciona como un multicliente BLE que hace de puerta de enlace LoRa a todas las notificaciones recibidas por los servidores/nodos conectados a él.*

Como se puede apreciar en el diagrama de flujos, existe un hilo principal que se centra en la configuración inicial de ambas comunicaciones, así como periféricos y dos callbacks para los eventos de dispositivo anunciado y notificación por parte de un dispositivo conectado.

Respecto a la programación del dispositivo, esta se ha realizado en Arduino debido a su mayor sencillez a la hora de programar. Por el contrario, sus librerías son mucho más limitadas y dependen directamente de la comunidad de

desarrolladores. En el caso del proyecto se partió de un ejemplo de cliente BLE [65] con capacidad de una conexión. Una topología de red ideal para el proyecto hubiese sido la topología en malla o Mesh, pero todavía se encuentra en desarrollo para esta familia de microcontroladores y los ejemplos publicados son muy escasos y poco documentados.

### 5.3.3 Fase 3: Comunicación Gateway-Backend Sigfox

El envío de información procedente de cada uno de los Gateway LoRa disponibles en cada vivienda será recibido por el ultimo dispositivo instalado en el sistema. Este dispositivo se trata de una lopy4 de la empresa Pycom, la cual, dispone de compatibilidad tanto con LoRa como Sigfox, tecnologías que intervienen en la capa de la arquitectura del sistema.

Respecto a su configuración y programación, los puntos más importantes son:

1. Toda la programación de este bloque se realizó en MicroPython con las librerías propietarias de Pycom. Este tipo de lenguaje es de alto nivel y muy fácil de desarrollar, sobre todo está orientado para el desarrollo de prototipos rápidos. También permite obviar todas las capas inferiores de dicha tecnología propietaria.

2. A pesar de que dispone compatibilidad en la recepción y transmisión de ambas tecnologías involucradas en el sistema, LoRa y Sigfox, el transceptor es el mismo. Por lo tanto, es necesaria la gestión a través de una máquina de estados para la habilitación y deshabilitación de cada uno de los sockets de comunicación según el estado de comunicación se encuentre.

3. La comunicación entre el Gateway de la vivienda y Gateway Sigfox se realiza a través de LoRa RAW, es decir, se está utilizando la capa de enlace del protocolo, pero no la capa de red. Por lo tanto, se puede considerar como una red LoRa privada que no usa los recursos de infraestructura de LoRaWAN.
   Uno de los inconvenientes más importantes de esta arquitectura es que el sistema receptor LoRa RAW, que estará en escucha continua (clase C), recibirá las tramas procedentes de todos los dispositivos transmisores, pertenecientes a la red del servicio o de otras redes. Por ello, es necesario el desarrollo de un sencillo protocolo de comunicación en la que cada trama irá empaquetada con una cabecera que permitirá al gateway receptor diferenciar los datos procedentes de la red que tiene asignada.

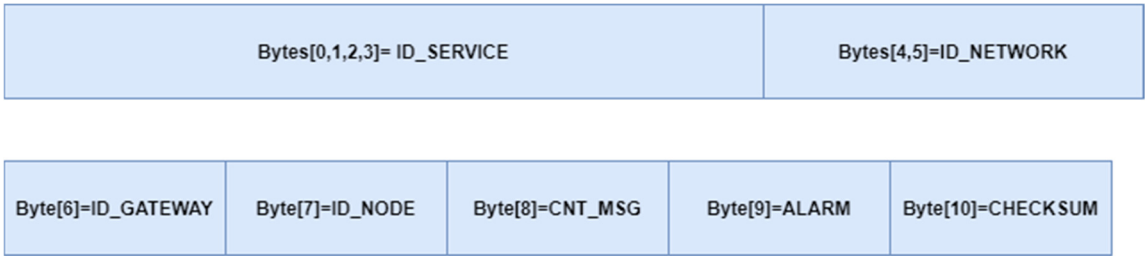



*Figura 40: Estructura de protocolo LoRa RAW.*

En la figura 40 se puede observar la trama diseñada para la transmisión y recepción de las tramas del sistema a través de LoRa RAW. Esta principalmente formada por:

- **Byte [3:0]= ID_SERVICE** representa a la cabecera que identifica al servicio. Todos aquellos mensajes que no tengan el valor asignado al servicio de la red serán descartados.

- **Byte [5:4]= ID_NETWORK** se trata del identificador de la red de cada sistema implantado. Cada conjunto de viviendas conectadas a un gateway Sigfox envían la información con este dato. Es simplemente informativo ya que podría existir la posibilidad de que no funcione un gateway de otro sistema, y estos sistemas de igual manera puedan transmitir a través de otro, lo que aporta una mayor robustez al sistema, a pesar de crear duplicidad en los datos.

- **Byte [6]=ID_GATEWAY** Identificador del gateway que envía el mensaje.

- **Byte [7]=ID_NODE** Identificador del nodo que envía el mensaje.

- **Byte [8]=CNT_MSG** contador de mensaje.

- **Byte [9]=ALARM** bandera que indica las alarmas que se han producido.
  0x01: Alarma de temperatura
  0x02: Alarma de gases
  0x04: Alarma de batería
  0x08: Alarma de pulsador

- **Byte [10]=CHECKSUM:** Para asegurar la correcta transmisión de la trama. Se transmite como último dato, el complemento a 2 de la suma de todos los valores anteriores de la trama de comunicación. Este valor se comprobará en destino para asegurar que la trama recibida es correcta.

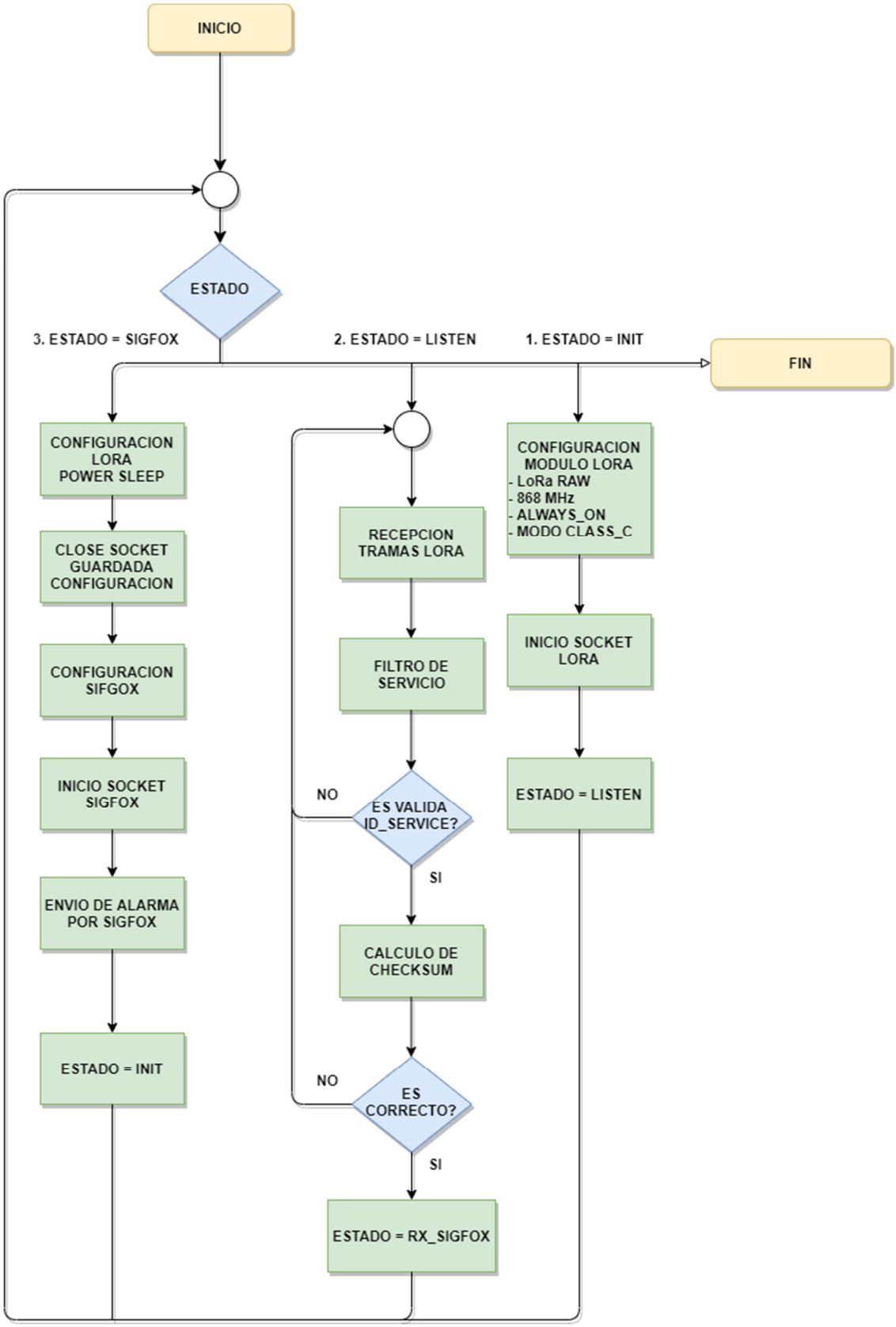



*Figura 41: Diagrama de flujo de firmware de Gateway Lora-Sigfox en Lopy4 de Pycom*
*El firmware se basa en una máquina de estados cuyo objetivo es escuchar en todo momento la recepción de posibles tramas del servicio, interrumpir su servicio LoRa y enviar la información a través de Sigfox, para finalmente volver a la escucha activa de más comunicaciones.*

### 5.3.4 Fase 4: Gestión de Callback y mensajes de dispositivos en la nube Sigfox

Después de la transmisión de las alarmas a través de Sigfox, estas, viajarán a través de la red propietaria del sistema hasta sus servidores dedicados. Aquí el usuario puede observar toda la información recibida. Pero antes de poder visualizar y tratar dicha información, es necesario registrar cada uno de los productos y pagar una suscripción por el mismo. El coste de la suscripción depende de la cantidad de información que se quiera enviar al día, el número de dispositivos a dar conectividad o la inclusión de servicios adicionales, como el de Atlas que se trata de un servicio de posicionamiento.

Después de realizar los pasos de suscripción a la red de nuestro dispositivo y registrarlo, se podrá acceder a toda la información enviada por el mismo, así como a opciones de configuración para el tratamiento de estos.

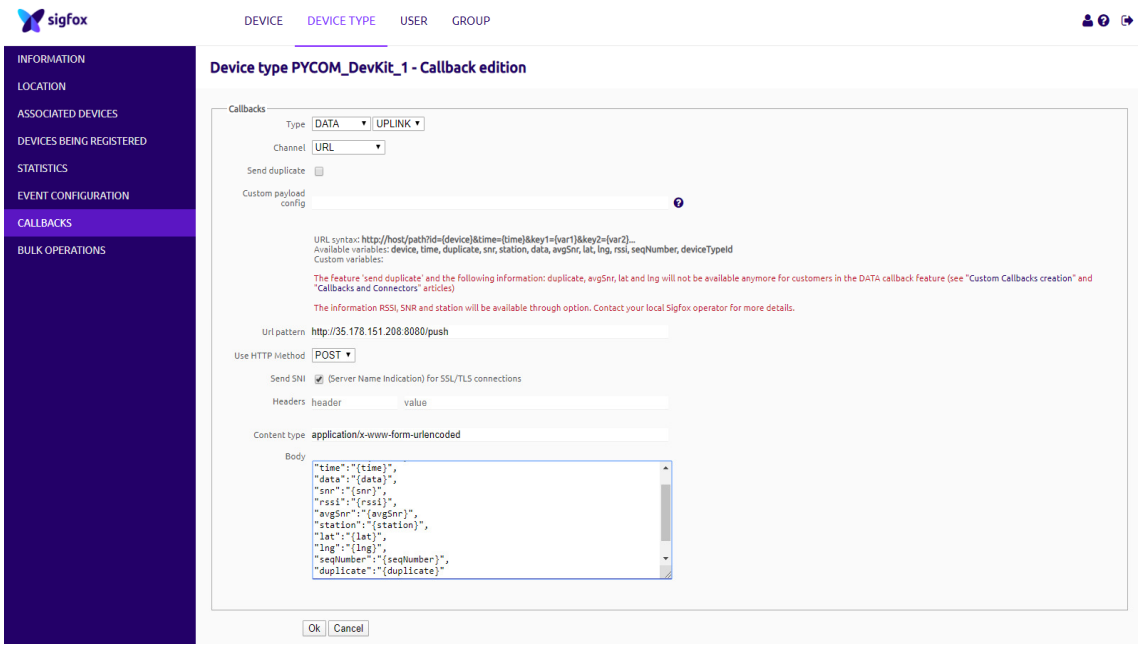



*Figura 42: Panel de control web para configuración de dispositivos y datos asociados Sigfox. En la imagen se puede observar las diferentes opciones, como localización, información cronológica de la recepción de datos y en particular la configuración y gestión de los datos recibidos*

Casi la totalidad de las opciones aportadas son de visualización de información, aunque existe una opción de configuración donde se puede indicar al servidor Sigfox a donde y como se quiere enviar la información recibida desde el dispositivo registrado.
Hay disponibles varias opciones [66] de como el usuario puede acceder a los datos:

- **Internet:** Se puede acceder a los datos de forma manual a través de la página web oficial de Sigfox, en ella se puede visualizar la información enviada por cada uno de los dispositivos registrados.

- **Callbacks:** Se trata de solicitudes HTTP (*Hypertext Transfer Protocol*), que son mensajes de notificación asociados a cada dispositivo registrado. Cuando el servidor de Sigfox recibe una transmisión por parte del dispositivo emisor asociado. De forma instantánea y con la configuración realizada previamente por el usuario, genera un mensaje para enviar la información de forma personalizada al servidor de cliente, de esta manera, el usuario no tiene que acceder de forma regular a través de la web al servicio. El único inconveniente de este servicio es que la información es unidireccional, y solamente se puede recibir información desde el servidor Sigfox, no pudiendo transmitir ordenes o respuestas al mismo y a su vez tampoco al dispositivo Sigfox asociado.
  Se pueden diferenciar de dos tipos. Callbacks personalizados en las que el usuario configura todos los parámetros, o propietarios en los que el usuario dispone de un servicio externo como AWS Dynamo o Azure que asocian la llamada a la plataforma del cliente.

- **API (***Application Programming Interface***):** En este caso se trata de solicitudes HTTP bidireccionales realizadas a través de una API. Por lo tanto, en este caso, sí que se permite el envío de respuestas al servicio Sigfox.
  La principal desventaja es que el usuario tiene limitado el número de solicitudes al servidor y por lo tanto es necesario seguir una serie de términos y condiciones **[67].**

- **Email:** También está disponible el envío de la información transmitida por el dispositivo vía email. Aunque este método es poco práctico a la hora de realizar una automatización de procesos, como guardado en BBDD o gestión por aplicaciones.

### 5.3.5 Fase 5: Servidor de cliente, almacenamiento y visualización de alarmas.

El último bloque del sistema y, por tanto, destino de toda la información generada por la red de sensores, es el servidor o plataforma de cliente. El objetivo de este bloque es la de recolectar, almacenar, listar y comparar toda la información recibida con la existente en su base de datos, para su posterior visualización al usuario final.

Para cumplir este cometido se ha creado una instancia a través del servicio lightsail de AWS **[68]**. Se trata de un servidor privado virtual o VPS (*Virtual Private Service*) que reside en la nube de Amazon. Esto permite disponer de la capacidad y funciones informáticas, de almacenamiento y de red para poder desarrollar y administrar aplicaciones web en la nube. También permite disponer de una IP estática para poder recibir sin ningún problema los datos enviados por los dispositivos de la red a través del frontend de Sigfox.

La principal razón de esta elección ha sido la problemática de conseguir una IP fija con una contratación de servicios de comunicaciones doméstica.
Actualmente la mayoría de las compañías de telecomunicaciones ofrecen

servicios con IP dinámicas, las cuales cambian constantemente si se reinicia el router de la vivienda. Actualmente la tendencia es incluso peor ya que las compañías de telecomunicaciones realizan configuraciones de red virtuales para que varios usuarios compartan una misma dirección IP o lo que es denominado la técnica CGNAT (*Carrier Grade Network Address Translation*), debido a este factor es imposible el envío de toda la información del servidor de Sigfox a nuestro equipo a través de un puerto abierto si la dirección del equipo no es accesible, además el uso de un servidor es una solución mucho más real con respecto a un servicio final.

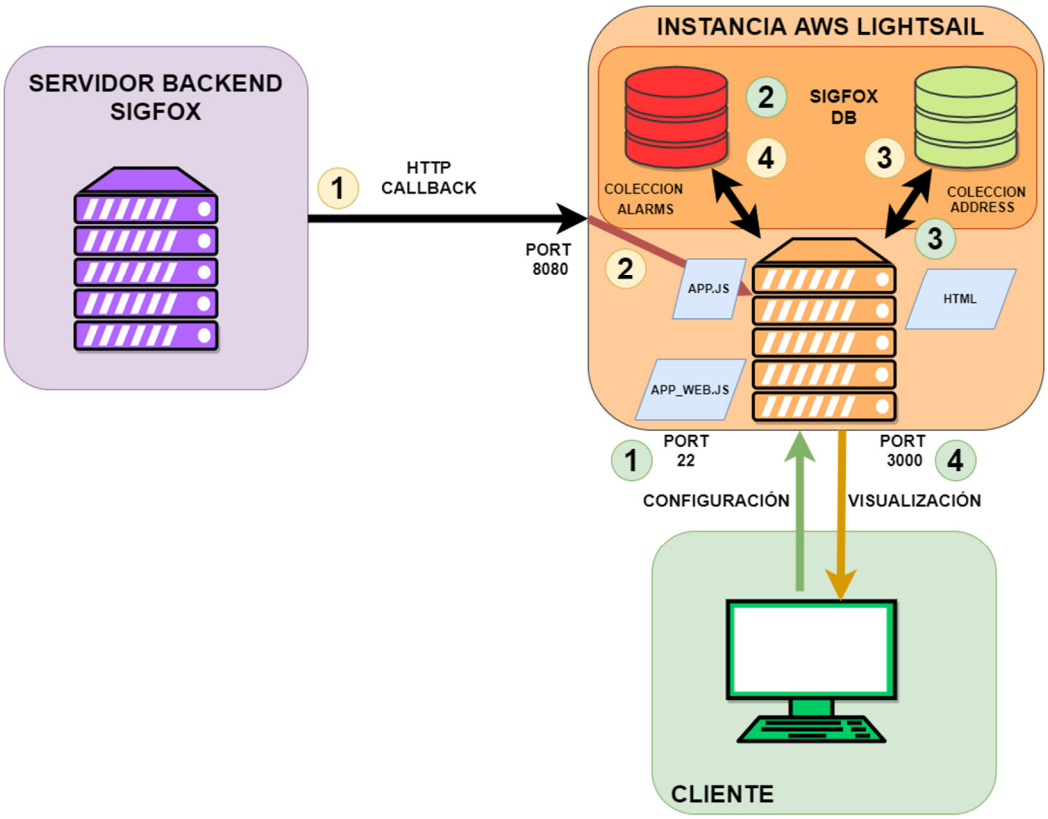



*Figura 43: Diagrama de sistema de emergencias basado en servidor virtual.*

En la figura 43 se puede observar el diagrama general de interacción entre los elementos finales del sistema de emergencias. Está formado por tres sistemas:

1. **Backend Sigfox:** Es la unidad que recibe la información enviada por el dispositivo. Este servidor la encapsula y envía según la configuración realizada por el usuario.

2. **Servidor/instancia AWS lightsail:** Alojamiento de todo el sistema de almacenamiento, procesamiento y visualización de alarmas. Dispone de una base de datos en mongodb con dos colecciones:
    - **Colección Alarms:** Colección de datos donde se almacenarán las alarmas recibidas. Dispone de los siguientes campos.

- **deviceId**: ID de dispositivo asociado a red Sigfox, será el identificador de red del edificio
- **date**: Fecha de recepción de alarma en el servidor
- **network**: ID de red de servicio
- **gateway**: ID de gateway que envío la alarma, este campo identifica en que piso se produjo la alarma.
- **Node**: Nodo que produjo la alarma
- **Count**: Número de mensaje producido por gateway
- **Alarm**: Valor de alarma producida
- **seqNumber**: Número de mensaje producido por el dispositivo Sigfox en el Callback.
- **address**: Dirección asociada a ID Device y gateway.

- **Colección Address**: Colección que contiene las direcciones asociadas a los dispositivos registrados en la red. Dispone de los siguientes campos:
    - **DeviceId**: Identificador de dispositivo Sigfox
    - **Gateway**: ID gateway de LoRa en la subred
    - **Address**: Dirección relacionada con los dos identificadores anteriores.

3. **Cliente:** Aplicación descargada a través de cliente web y ejecutada de forma local por el usuario.

Estos tres elementos realizan dos series de interacciones entre ellos. En la figura 43 se pueden observar en círculos dos secuencias con diferente color, representando ambos procesos.

En el primer proceso (círculo amarillo) intervienen backend de Sigfox y servidor de cliente. El servidor de cliente es el encargado de escuchar de forma permanente el puerto 8080 al envío de tramas procedentes del backend de Sigfox. Tras la recepción de una trama valida, se extraen los datos y con los campos de “devideId” y “Gateway” se consulta en la colección address si existe alguna posición coincidente, de ser así el campo dirección es copiado y almacenado con el resto de los datos extraídos, en la colección alarmas para una posterior consulta.

En el segundo proceso (circulo verde) intervienen el servidor de cliente y el equipo final de usuario. En esta comunicación, el usuario accede al servidor a través de un cliente web enviando una petición a la dirección IP del servidor en el puerto 3000. Tras esta primera solicitud, recibe la página principal de gestión de emergencias donde se pueden realizar diferentes tareas sobre la base de datos almacenada. En la siguiente figura se puede observar el panel de control.

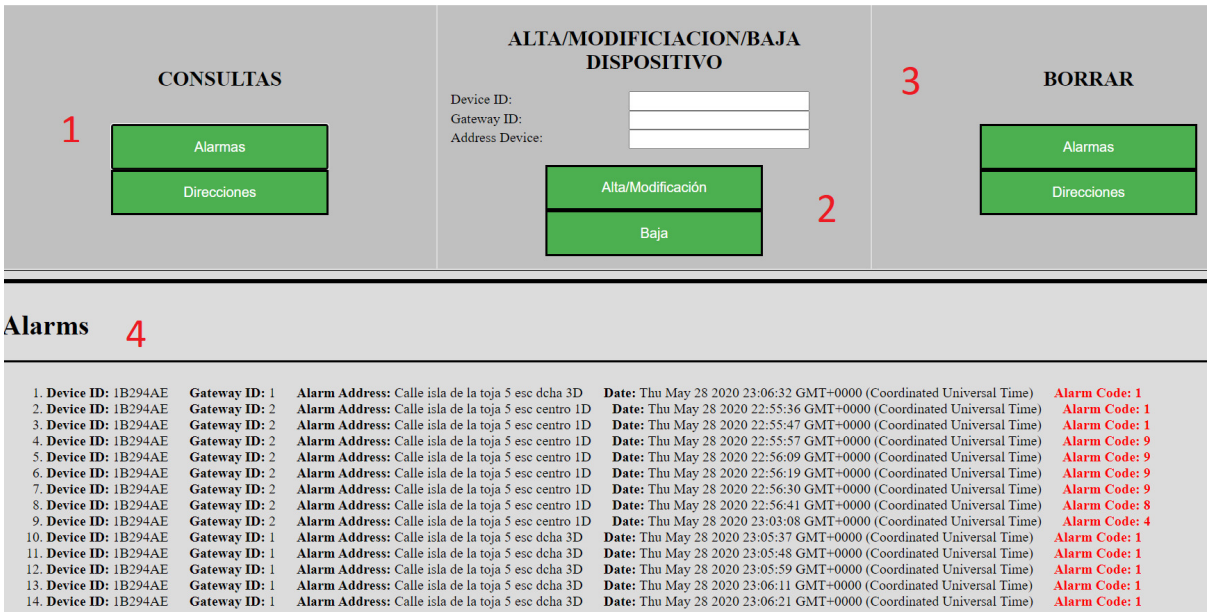



*Figura 44: Panel de control de sistema de detección de emergencias.*

Esta aplicación se caracteriza por un panel superior donde se dispone de todas las acciones posibles (1,2,3), mientras que la parte de control (4) es la consola donde se puede observar todos los resultados a las peticiones. La configuración se ha realizado en HTML(*HyperText Markup Language*) [69] y JavaScript [70] con jQuery.ajax (*Asynchronous JavaScript And XML*) [71] permitiendo realizar solicitudes desde una aplicación almacenada en el cliente web de forma local mientras que las diferentes respuestas a las solicitudes se actualizan solamente en la parte inferior.
Las configuraciones y solicitudes disponibles son:

1. **Consultas:** Permite la visualización de las alarmas y direcciones almacenadas en la base de datos. Además, cuando se realiza la petición de alarmas sin realizar ninguna acción posteriormente, permite la visualización y recepción de las alarmas en tiempo real (un retardo máximo de 2 segundos) gracias a que se refresca automáticamente.

2. **Alta/modificación/baja de dispositivo**: Permite al usuario realizar gestiones sobre un dispositivo pudiendo dar de alta un dispositivo/gateway y asociar a este una dirección, de tal manera que cuando se produzca una alarma, el usuario disponga de la información detallada de la vivienda sin realizar ninguna consulta manual. Cuando se introduce un “Device ID” y “gateway ID” existentes y se introduce una dirección diferente, esta será actualizada pulsando el botón de alta/modificación o eliminada de la base de datos pulsando el botón de baja.

3. **Borrar**: Permite limpiar todas las posiciones existentes tanto en la colección de alarmas como en la colección de direcciones.

*Todo el código fuente, así como el tutorial de configuración del servidor se pueden encontrar en los ANEXOS C y D de la memoria del trabajo.*

# 6. Pruebas sobre el sistema completo y análisis de los resultados.

## 6.1 Configuración de sistema para pruebas.

A continuación, para realizar las pruebas sobre un caso real del sistema propuesto, se plantea la siguiente configuración de red. En esta configuración del sistema se propone una simulación de red completa de dos viviendas, tal y como se indica en la figura 45. En cada una de ellas estarán posicionados dos nodos y un gateway LoRa y en la azotea del edificio se estacionará el gateway Sigfox para asegurar una mayor cobertura.

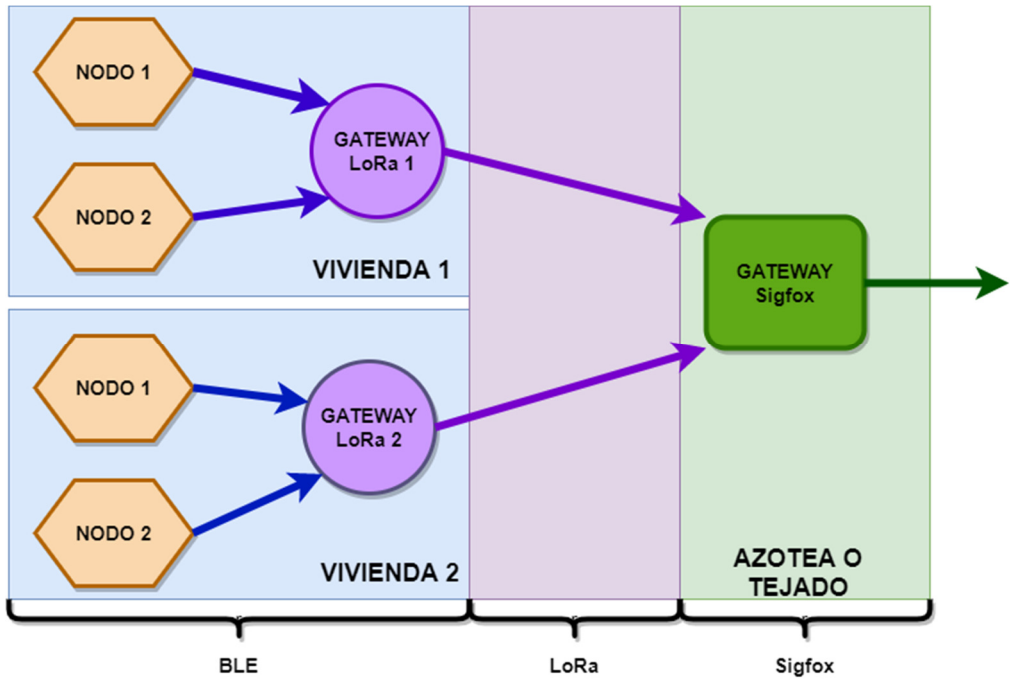



*Figura 45: Diagrama de arquitectura propuesta para pruebas reales*

La red WPAN dedicada para la vivienda 1 estará formada por los dispositivos diseñados y montados para este trabajo, mientras que los elementos que formarán la red de la vivienda 2 serán los prototipos desarrollados a partir de las placas comerciales de pruebas adaptadas para este propósito. Además, debido a que estarán posicionados de forma autónoma y sin conexión alguna, para comprobar su estado, el gateway tiene integrado una pantalla OLED que indica el número de dispositivos conectados y mostrará la información de notificaciones que pase a través de él.

A la hora de realizar la configuración de cada firmware es necesaria la personalización de alguno de sus parámetros para poder identificar correctamente a cada nodo. Estos parámetros son:

- **NODE_NAME**: Nombre de nodo que se visualizará cuando se anuncie a través de BLE

- **NODE_ID**: Identificador de nodo dentro de la red WPAN

- **SERVICE_UUID**: Identificador único del servicio de subred BLE de la vivienda.

Respecto a los umbrales de detección de alarma se han establecido unos que no corresponden con un caso real pero que son fácilmente reproducibles como, por ejemplo, establecer como umbral de temperatura, 30ºC, lo suficiente para poder activar la alarma con el contacto del cuerpo humano, en un caso real este umbral estaría aproximadamente entre los 50º y 60ºC. Los umbrales establecidos para niveles de gas será un valor de IAQ 420 y el número de veces que se deberá pulsar de forma consecutiva es de 5 veces.

La configuración del gateway, también es necesaria la edición de parámetros y para una correcta identificación de la red y de los dispositivos dentro de la arquitectura. Estos parámetros son:

- **ID_GATEWAY:** Es el identificador que irá dentro del paquete de la trama enviada al gateway Sigfox y que será esencial para la detección de la vivienda donde ha ocurrido la emergencia

- **serviceUUID**: Identificador único del servicio de subred de la vivienda. Si no son iguales en el nodo y en el Gateway, este último no se conectará a los nodos a pesar de que escanee y visualice los anuncios de estos.

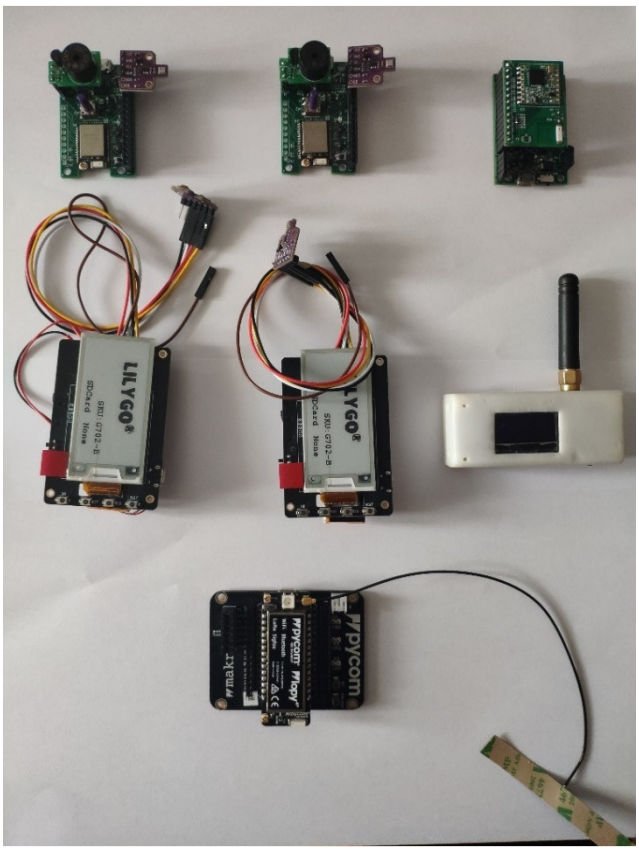



*Figura 46: Elementos utilizados para la prueba del sistema*

Otros puntos que resaltar sobre el sistema son:

1. El firmware no está en modo bajo consumo, la conexión BLE entre servidores y cliente se establece y se mantiene mientras los equipos están encendidos.

2. El módulo BME680 diseñado para el prototipo final, no ha podido utilizarse debido a la imposibilidad de adquirir el integrado. En su lugar se ha utilizado una placa genérica que es compatible con la configuración de pines de la placa.

### 6.2 Primeras pruebas sobre sistema cableado.

El primer análisis de sistema se trata de la conexión de todos los sistemas al ordenador a través de USB para poder visualizar a través del puerto serie, los mensajes de consola generados, de esta manera se podrá realizar un primer análisis del funcionamiento del sistema.

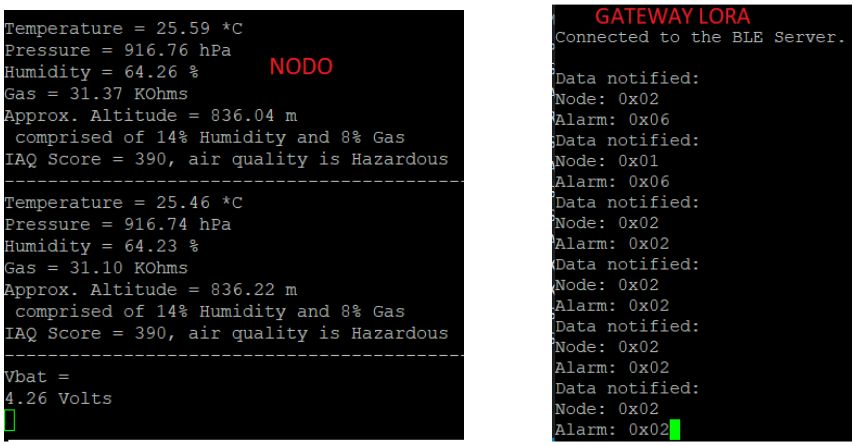



*Figura 47: Arranque de red WPAN, visualización de consola por puerto serie*

En la figura 47 se puede observar el proceso de encendido del dispositivo. Se puede apreciar que se produce una serie de alarmas causadas por notificaciones de ambos nodos al calcular el IAQ. Esto es debido al tiempo de respuesta del propio sensor al realizar mediciones del aire. En este caso nada más encender el sensor dará valores falsos, hasta que, pasado un tiempo, del orden de minutos, el sensor empiece a tomar medidas correctas. En este aspecto, esta prueba da una pauta importante a la hora de instalar el sistema y es que es necesario el encendido previo de los nodos para una estabilización de las medidas del sistema.

Respecto al arranque del gateway LoRa, nada más arrancar realiza un escaneo de los dispositivos BLE disponibles y correctamente establece conexión con aquellos que pertenecen a su subred, es decir, aquellos servidores que disponen del mismo UUID de servicio.

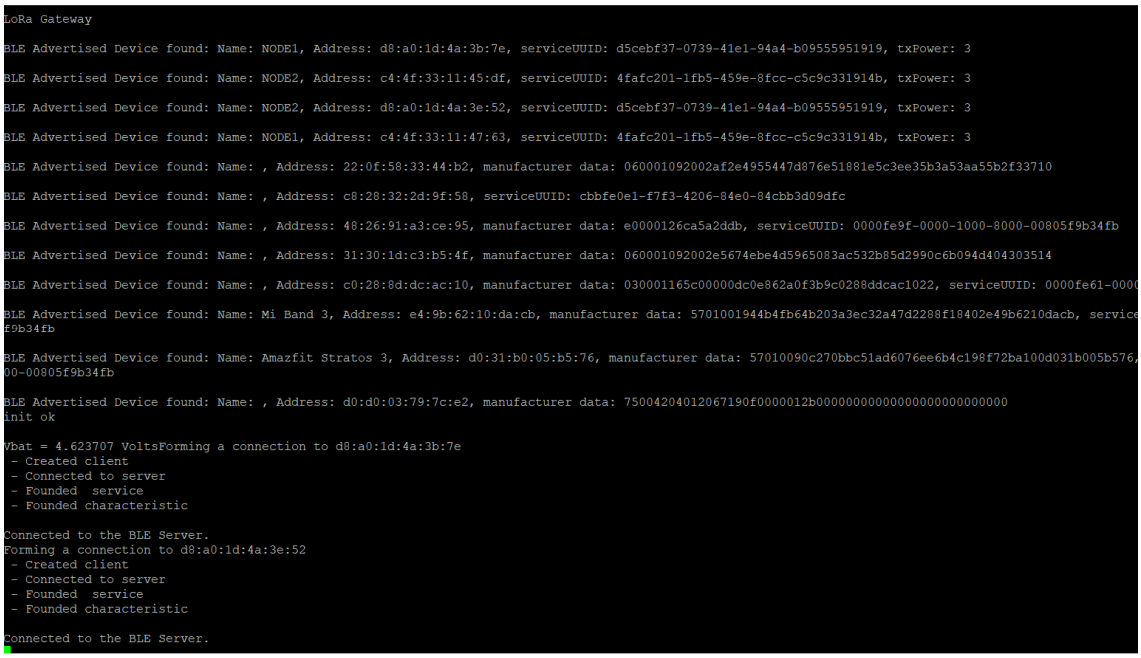



*Figura 48: Escaneo de dispositivos por parte del gateway LoRa y proceso de conexión.*

En la figura 48, se puede apreciar como durante el escaneo de dispositivos detecta todos los dispositivos bluetooth dentro de su cobertura, pero solamente conecta con aquellos que dispongan del mismo identificador de servicio.

Respecto al propio funcionamiento de los sistemas fabricados hay que destacar tres problemas de diseño o montaje:

1. Para conseguir que el sistema funcione bajo modo Deep sleep es necesario conectar el GPIO16 al pin de reset para poder despertar y reiniciar el circuito en el tiempo configurado. Este elemento debe ser configurable, ya que, durante la descarga del firmware, esta conexión no debe existir.

2. El sistema, al conmutar el interruptor deslizante de encendido, no arrancaba el firmware correctamente y era necesario el uso del botón pulsador de reset posteriormente para iniciar el funcionamiento del sistema. La razón de este comportamiento incorrecto fue el montaje en el circuito de arranque un condensador de 1uF en el pin IO0 encargado de la activación de modo descarga del microcontrolador. Eliminado este condensador el dispositivo funciona correctamente.

3. El botón de emergencias estaba rutado inicialmente al pin de entrada I36. Al realizar el arranque del sistema se apreció de que, sin pulsar el botón, el sistema detectaba continuamente la pulsación de este a través de la interrupción. Tras investigar, se concluyó que no es posible el uso de este pin cuando simultáneamente se está usando el ADC1 del microcontrolador (este ADC1 se usa para medir el nivel de la batería). Para solucionar el problema se cableo el circuito del pulsador al pin I34.

### 6.3 Pruebas del sistema completo

A continuación, se realizará una prueba del sistema completo simulando dos alarmas diferentes, la primera por temperatura y la segunda mediante la pulsación del botón de emergencias.

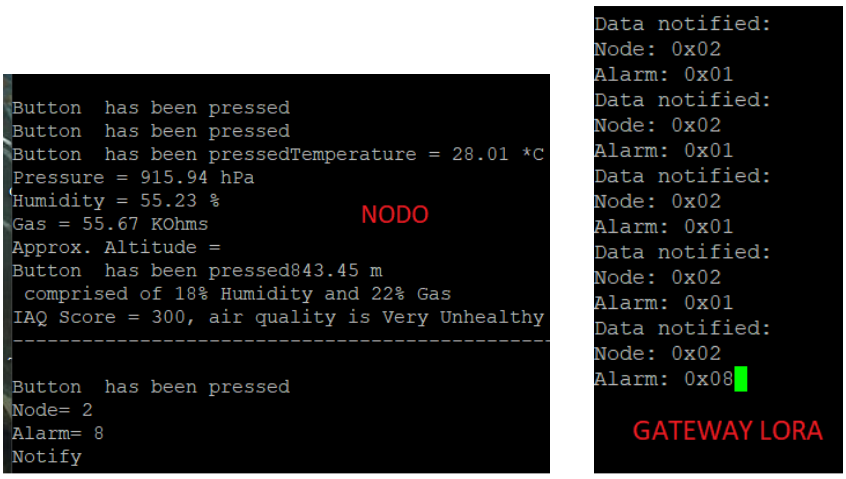



*Figura 49: Proceso de envío de alarma. Parte 1. Dispositivos del Edificio*

En la figura 49 se puede apreciar como en el nodo se detectan una serie de pulsaciones en el botón de emergencia, produciendo una notificación de alarma con un valor de 0x08, esta trama compuesta por dos bytes que contiene el número de nodo y el tipo de alarma enviado. El gateway LoRa recibe la notificación de la alarma y genera una trama con la cabecera del servicio de emergencias para ser enviada al gateway Sigfox que está en continua escucha.

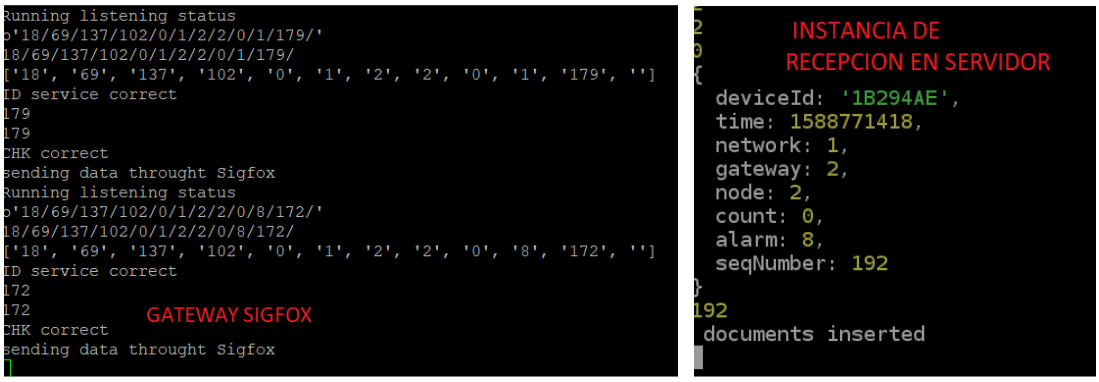



*Figura 50: Proceso de envío de alarma. Parte 2. Recepción en servidor*

En la imagen izquierda de la figura 50 se puede observar la recepción de la alarma por parte del gateway Sigfox. Este dispositivo realiza un análisis de la trama, comprobando que el ID del servicio es el correcto. Inmediatamente después genera un checksum y verifica que la comunicación se realizó correctamente. Después, la información es enviada a través de la red Sigfox hasta su backend. Tras la recepción del dato por parte del backend, el servidor Sigfox detecta que existe un Callback asociado a ese dispositivo, por lo que, siguiendo las instrucciones de la configuración, la trama es reenviada al

servidor del cliente. En la imagen de la derecha de la figura 50 se puede observar la recepción, extracción y almacenamiento de la información por parte del servidor virtual. Esta información será almacenada en la base de datos en mongo asociada a las alarmas.

*La codificación de la instancia de recepción se puede observar en el ANEXO C del documento.*

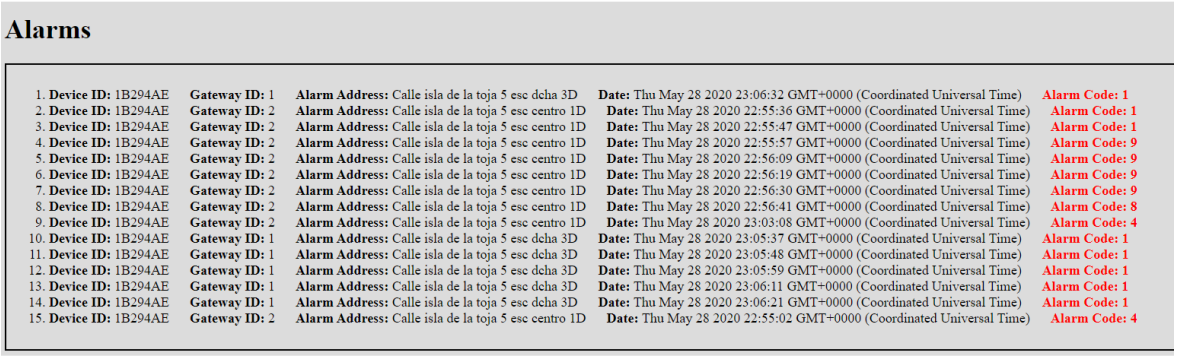



*Figura 51: Proceso de envío de alarma. Parte 3. Listado de alarmas almacenadas*

Simultáneamente, se estará ejecutando otra instancia (escuchando un puerto diferente) que escuchará peticiones por parte del usuario para listar las alarmas producidas y visualizarlas en el explorador. Esta lista se irá refrescando periódicamente para evitar posibles retrasos en la respuesta del servicio ante una posible emergencia.
Todo el proceso, desde la superación del límite de la alarma, hasta la visualización por parte del usuario tuvo un retardo sorprendentemente bajo, del orden de unos 3 segundos, lo que permite una reducción considerable en el tiempo de respuesta por parte del centro de emergencias que recibe la alarma.

### 6.4 Pruebas de medición de temperatura.

Tras la realización de varias pruebas sobre la fiabilidad de comunicación en el sistema, se detectó que los diferentes nodos realizaban mediciones dispares. Aquellos dispositivos que fueron diseñados incluyendo la placa de circuito impreso presentan un offset en la temperatura de 3ºC, la conclusión a la que se llegó es que al estar directamente insertados encima de la placa principal sufren un ligero autocalentamiento por la temperatura disipada por el circuito principal.

Una posible solución es el uso de cableado para colocar dicho sensor en una posición más alejada del circuito y así evitar este fenómeno, tras varias pruebas con esta solución se pudo observar que los valores de todos los dispositivos se mantuvieron en un margen de dispersión de +-1ºC.

Aun así, un offset de 3ºC no es un error tan importante debido a la propia naturaleza del sistema, que es la detección de temperaturas mucho mayores, por lo que se puede mantener la arquitectura modular planteada y aceptar ese error de temperatura.

### 6.5 Pruebas de medición de gases.

A pesar de que el parámetro de medida principal del sistema es la temperatura, la posibilidad de detectar gases puede ser un buen recurso como parámetro complementario. El sensor BME680 dispone de una resistencia que da un valor óhmico según la concentración de gas pueda detectar. Por otro lado, existe un parámetro denominado IAQ o Indoor Air Quality que indica la calidad de aire a través de un complejo algoritmo y a partir de dos parámetros, la humedad relativa y el valor resistivo variable con la concentración de los gases sensibles.

En el caso de este trabajo es mucho más importante la detección de gases y por lo tanto no se puede establecer dicho parámetro con la fiabilidad que debería si se incluye la humedad relativa como un parámetro de peso en esta medida, un ejemplo de posible falso positivo sería un habitáculo con mucha humedad, una cocina, donde se puede desprender gran cantidad de vaho, un valor de humedad relativa del 100% también podría considerarse un valor IAQ pésimo para la calidad del aire en una zona cerrada.

Por lo tanto, a partir de un código en Arduino [72] se realizarán una serie de modificaciones en el código para cambiar el peso de los parámetros a la hora de calcular la calidad del aire y posteriormente una serie pruebas para corroborar la fiabilidad de la medición de altas concentraciones de gases.
En esta serie de pruebas se realizará sobre los gases dióxido de carbono y vapores emanados por líquidos industriales como acetona.

Al iniciar el análisis sobre ambos nodos, se aprecia que los valores de gases son demasiado altos, esto como se explicó anteriormente es debido a que es necesario un tiempo de estabilización en los valores de medición de gases medidos. Después de un considerable periodo de tiempo (2 horas) se puede apreciar un descenso y estabilización del valor de lectura. Consiguiendo que los nodos aportasen valores IAQ similares. (hay que destacar que este periodo de tiempo difiere de unos a otros módulos de sensores) Por lo que ya se puede proceder a la prueba de gases.

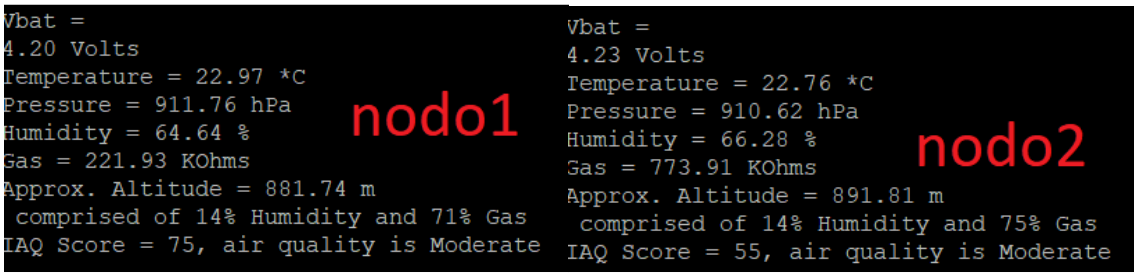



*Figura 52: Comparativa medición de gases. Medición factor IAQ entre dos nodos.*

Se puede apreciar en la figura 52 que el algoritmo IAQ a pesar de que detecta valores diferentes en la resistencia de gases, consigue valores muy similares en este parámetro, posiblemente debido a las calibraciones de fábrica de cada uno de ellos. También puede ser una de las razones por las que difiere el tiempo de estabilización de este parámetro.

| Casos | | IAQ | R (kΩ) |
|---|---|---|---|
| **Normal** | **Nodo1** | 75 | 209 |
| | **Nodo2** | 55 | 692 |
| **CO2** | **Nodo1** | 350 | 68,67 |
| | **Nodo2** | 110 | 239,74 |
| **Acetona** | **Nodo1** | 425 | 6,34 |
| | **Nodo2** | 300 | 111,7 |

*Tabla 8: Comparativa de medida de gases entre diferentes sensores*

Tras someter a altos valores de dióxido de carbono a ambos sensores, en uno de ellos el valor IAQ se disparó hasta los 350, mientras que en el otro nodo solamente se incrementó hasta los 110 aun así se mostró una reducción considerable de la resistencia de medición de gases.

La segunda prueba a la que se sometieron los nodos fue con acetona [73], un material disolvente utilizado como limpiador industrial. La respuesta de ambos sensores fue parecida, se mostró una reducción en el valor resistivo interno, y por lo tanto un valor mayor de IAQ, lo que indica una pérdida de calidad del aire. Aunque los valores son más parecidos que en el caso de CO2 existe una gran disparidad entre ambos valores, lo que sugiere que el sistema requiere un mayor estudio para conseguir una mayor fiabilidad.

### 6.6 Pruebas de consumo.

A continuación, se realizan una serie de pruebas de consumo sobre los dispositivos diseñados. La medida del consumo en el dispositivo en Deep sleep es imposible de medir debido a la resolución de la instrumentación de medida disponible. Aun así, si se pueden realizar medidas sobre los dispositivos en pleno funcionamiento. Para ello se propone el siguiente diagrama de montaje para la medida de la corriente que es capaz de requerir el circuito.

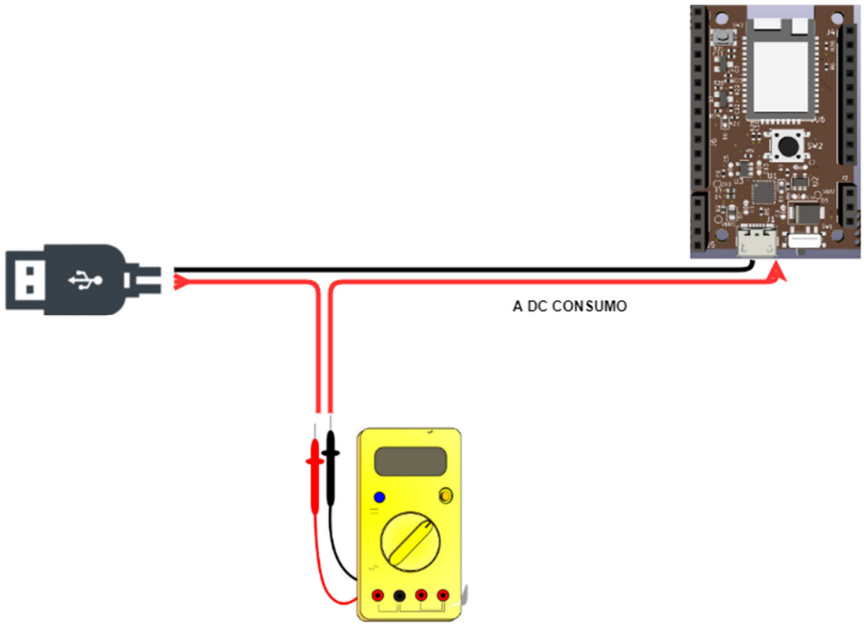



*Figura 53: Diagrama de medición de consumo sobre dispositivos*

Se puede observar en la figura 53, el esquema para la medida de la corriente que circula por el circuito. El único accesorio por fabricar es el cableado de alimentación. Este elemento se puede obtener a partir de un cable USB comercial y cortar el cable positivo de alimentación para dejarlo en circuito abierto. Las bornas del polímetro se conectarán a ambos extremos del corte pudiendo circular la corriente requerida por el circuito y por tanto poder cuantificar la corriente que pasa a través de él.

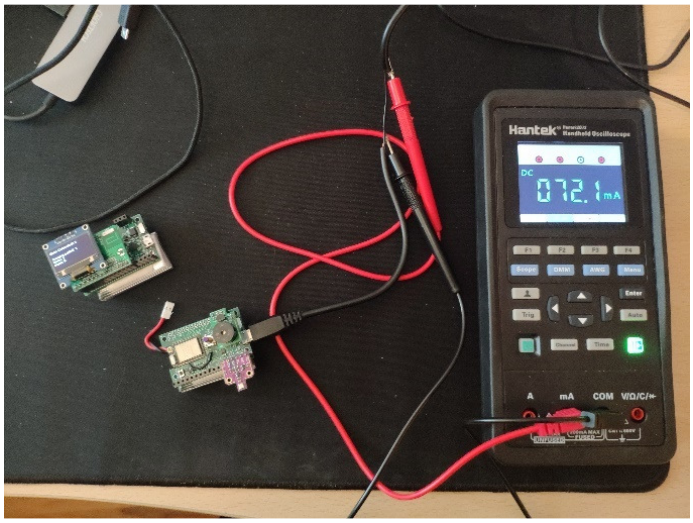



*Figura 54: Pruebas de consumo sobre nodo/sensor*

Para este tipo de medida se tuvo que desconectar la batería para evitar una medida falsa del consumo real del dispositivo.

Los resultados obtenidos son:

| **Dispositivo** | **Corriente** |
|---|---|
| Nodo/sensor (BLE+BME680 full measure) | 70 mA inicialmente, tras configurar la frecuencia CPU a 80 MHz bajo a 50 mA |
| Nodo/sensor (Alarma Sonora ON) | 120mA |
| Gateway (Escucha BLE) | 29 mA |
| Gateway (Transmitiendo LoRa) | 40 mA |

*Tabla 9: Consumo de dispositivos en situaciones de comunicación y medida.*

Se puede apreciar que el consumo en los picos de funcionamiento es superior a los estimados en los cálculos teóricos. Los elementos diferenciadores en ese consumo entre ambos dispositivos son:

1. El consumo del sensor BME680 es superior: Esto es debido a la alta tasa refresco en la medición, la medida de parámetros como la humedad relativa y el mayor consumo que supone la medida de gases.

2. El nodo al funcionar como servidor BLE y encargarse de la transmisión, tiene un mayor consumo que él sistema receptor BLE según indican las especificaciones mostradas en la tabla 4 del trabajo.

## 6.7 Análisis de resultados.

Tras las pruebas realizadas al sistema se puede concluir que los resultados no son tan buenos como se preveía en los cálculos teóricos.

1. **Medición de temperatura:** El único punto que podría considerarse totalmente satisfactorio es que tras las pruebas de medición de temperatura realizadas podemos decir que se trata de un sistema bastante exacto y preciso. Al realizar las pruebas, los 4 nodos aportaron valores muy cercanos unos a otros, por lo que la dispersión de las medidas era bastante baja. También las medidas tomadas con respecto a otras estaciones meteorológicas comerciales eran muy parejas.
   El único inconveniente que presento fue un problema de autocalentamiento en los dispositivos diseñados debido a la posición del módulo respecto a la placa principal. Al estar tan cerca el sensor, este era sometido a un incremento de temperatura de +3ºC, pero al posicionarlo a una mayor distancia gracias a un cableado entre placa principal y modulo sensor, la temperatura descendió hasta valores similares a los tomados por los otros prototipos. Aun así, se consideró un problema menor ya que no es necesario un sistema preciso para la detección de incendios.

2. **Medición de gases:** La medición de gases podría ser un sistema complementario a la medición de temperatura. Se ha comprobado que el sensor es realmente sensible ante la exposición a distintos gases. Aun así, existen muchas diferencias entre las medidas tomadas entre los nodos y requiere de un mayor estudio para conseguir un funcionamiento fiable y robusto.

3. **Consumo:** Tras haber realizado varias pruebas de consumo sobre todas las configuraciones del sistema (nodos y gateway), se ha podido observar una desviación en los valores reales medidos con respecto al consumo estimado teóricamente, lo que provocará una reducción en la autonomía en los dispositivos. Este incremento puede ser debido a:
   a. una mayor tasa de medición realizada durante las pruebas (1 medida cada 10 segundos),
   b. medidas de humedad relativa y presión además de muestreo de temperatura y gases
   c. un mayor consumo la medida de gases que el estimado teóricamente.

4. **Rendimiento del sistema:** En general el rendimiento ha sido satisfactorio electrónicamente. Aun así, de seguir con el sistema presentado en el trabajo, sería necesaria una revisión de la electrónica para corregir los fallos detectados que afectan al arranque del sistema en modo Deep sleep y a la detección de la pulsación del botón de emergencias.

# 7. Valoración económica del trabajo

En el siguiente capítulo se presentará un análisis y valoración de todos los costes asociados al desarrollo del proyecto, así como una estimación del coste de producción de este.

### 7.1 Costes asociados al desarrollo del prototipo.

| Referencia | Descripción | PVP | Cantidad | Total |
|---|---|---|---|---|
| **Mouser** | | | | |
| CMT-1285C-035 | Avisador acústico piezoeléctrico | 0,792 € | 1 | 0,792 € |
| CP2104-F03-GM | Conversor USB-UART | 1,490 € | 1 | 1,490 € |
| AP2112K-3.3TRG1 | Reguladores de voltaje LDO | 0,423 € | 1 | 0,423 € |
| MCP73832T-2ACI/OT | IC gestor de baterías | 0,504 € | 1 | 0,504 € |
| 2N5550TAR | Transistor bipolar BJT NPN | 0,189 € | 1 | 0,189 € |
| KMR421NGLFS | Interruptor Táctil | 0,401 € | 1 | 0,401 € |
| APG1608QBC/D | Led ámbar 0603 | 0,130 € | 1 | 0,130 € |
| JMK105BJ105KP-F | Condensador 1u 0402 | 0,041 € | 4 | 0,164 € |
| CRCW0402100KFKEDC | Resistencia, 100k 0402 | 0,029€ | 3 | 0,870 € |
| CRCW040210K0FKEDC | Resistencia, 10k 0402 | 0,029€ | 8 | 0,232 € |
| MMBT3904,215 | Transistor bipolar npn SOT23-3 | 0,078€ | 2 | 0,176 € |
| 620102131822 | Alojamiento cable 2mm | 0,960 € | 1 | 0,960 € |
| GRM188R60J106ME84J | Condensador 10uF 0603 | 0,201€ | 6 | 1,206 € |
| MBRS2040LT3G | Diodo Schottky SMB | 0,324 € | 1 | 0,324 € |
| 10118192-0001LF | Conector MicroUSB | 0,333 € | 1 | 0,333 € |
| 929974-01-26-RK | Alojamiento cable | 2,430 € | 1 | 2,430 € |
| 93992-436HLF | Alojamiento cable | 0,648 € | 1 | 0,648 € |
| CRCW04020000Z0EDC | Resistencia 0R 0402 | 0,029 € | 5 | 0,145 € |
| CRCW04024K70FKEDC | Resistencia 4K7 0402 | 0,029 € | 3 | 0,870 € |
| CRCW04021K00FKEDC | Resistencia 1k 0402 | 0,029 € | 1 | 0,029 € |
| CRCW040222R0FKEDC | Resistencia 22R 0402 | 0,029 € | 6 | 0,174 € |

| | | | | |
|---|---|---|---|---|
| CRCW0402180RFKEDC | Resistencia 180R 0402 | 0,029 € | 1 | 0,029 € |
| LL3301DF065QJ | Pulsador 13 mm | 0,540 € | 1 | 0,540 € |
| BME680 | Sensor medioambiental | 8,290 € | 1 | 8,290 € |
| SD0805S020S0R5 | Diodo Schottky 20V 0.5A | 0,396 € | 1 | 0,396 € |
| 0868AT43A0020E | Antena 868MHz Cerámica | 0,837 € | 1 | 0,837 € |
| PCM12SMTR | Conmutador deslizante | 0,810 € | 1 | 0,810 € |
| **Aliexpress:** | | | | |
| SX1276 MOD | Modulo sx1276 ISM | 4,90€ | 1 | 4,90€ |
| TTGO micro-32 | Microcontrolador SoM ESP32 | 3,78€ | 1 | 3,78€ |
| **JLCPCB:** | | | | |
| PCB BUZZER | PCB 2 capas | 0,73€ | 1 | 0,73€ |
| PCB SENSOR | PCB 2 capas | 0,366€ | 1 | 0,366€ |
| PCB LORA | PCB 4 capas | 2,374€ | 1 | 2,374€ |
| PCB PRINCIPAL | PCB 4 capas | 2,374€ | 1 | 2,374€ |
| **Amazon:** | | | | |
| B078WQ1694 | Batería Litio 600 mA | 4,50€ | 1 | 4,50€ |
| **TOTAL (SIN IMPUESTOS)** | | | NODO = 29,805€ | GATEWAY= 27,198€ |
| **IMPUESTOS (21%)** | | | 6,26€ | 5,71€ |
| **GASTOS DE IMPORTACION (DUA+ARANCEL)** | | | - | - |
| **TOTAL (IVA incl)** | | | **40,565€** | **37,408€** |

*Tabla 10: Coste de lista de materiales de prototipos desarrollados.*

La tabla 10 muestra los costes asociados a la fabricación de placa de circuito impreso y componentes para el montaje manual de un prototipo. Aquellos que están subrayados en verde son los elementos asociados exclusivamente al montaje de un nodo, en amarillo los elementos exclusivos a fabricación de gateway. Otros elementos por destacar:

1. La mayoría de los pedidos se realizaron a través del distribuidor Mouser [74], permitiendo el ahorro de los gastos de envío y los costes de importación.

2. Tanto microcontrolador como modulo LoRa se realizaron a través de Aliexpress, al no superar el coste mínimo de importación, estos elementos están exentos de gastos de aduanas (DUA y Aranceles).

3. El coste total debe incluir el coste de mano de obra de montaje, así como el material para 1 gateway y dos nodos. El cálculo de la mano de obra necesaria para la soldadura de los componentes se ha calculado a partir del sueldo bruto estipulado en el convenio colectivo del sector de empresas de ingeniería y oficinas de estudios técnicos de 2019 [75] por un ingeniero de nivel 1 en 2020. En el cálculo se ha asumido los costes

que repercuten desde el lado de la empresa, por lo tanto, es necesario incluir un 30% de costes asociados a cotizaciones de la seguridad social.

La fórmula de coste/hora es:

$$\textbf{\textit{Coste Mano de Obra}}\left(\frac{€}{\textbf{\textit{hora}}}\right) = \frac{\textbf{\textit{salario bruto Convenio}} + 30\%\ \textbf{\textit{Cotizacion}}}{\textbf{\textit{Numero horas jornada ordinaria anual}}} =$$

$$= \frac{26323,57 + (26323,57 \cdot 0,3)}{1792} = 19,1 \frac{€}{\textbf{\textit{hora}}}$$

| Concepto Coste | Coste (€/ud) | unidades | Total (€) |
|---|---|---|---|
| **Componentes prototipo** | | | |
| Nodo | 40,56 € | 2 | 81,13 € |
| Gateway | 37,41 € | 1 | 37,41 € |
| **Componentes comerciales (Aliexpress)** | | | |
| Lilygo T5 | 11,98 € | 2 | 23,96 € |
| Lilygo T3 Lora32 | 14,54 € | 1 | 14,54 € |
| Módulos BME680 | 9,35 € | 4 | 37,40 € |
| **Componentes comerciales (Pycom)** | | | |
| Pycom Lopy4 | 34,95 € | 1 | 34,95 € |
| Expansión Board Pymakr v3 | 16,00 € | 1 | 16,00 € |
| **Otros:** | | | |
| Batería LiPo 650mA | 4,50 € | 4 | 18,00 € |
| Montaje | 19,10 € | 12 | 229,00 € |
| Impuestos | 13,00 € | 1 | 13,00 € |
| Envío | 14,00 € | 1 | 14,00 € |
| **Total** | | | **519.4€** |

*Tabla 11:Coste total de sistema prototipo para el trabajo*

En la tabla 11 se puede observar todos los gastos asociados al desarrollo de la parte hardware del sistema propuesto en el trabajo.
En la tabla se incluye:

1. Coste de dispositivos diseñados
2. Coste de dispositivos prototipos a partir de elementos comerciales
3. Mano de obra asociada a su fabricación

### 7.2 Coste asociado a Ingeniería.

| Concepto | Precio(€/hora) | Unidades | Total |
|---|---|---|---|
| Coste ingeniería | 19,1 | 400 | 7640 € |

*Tabla 12: Costes asociados a gastos de ingeniería.*

En la tabla 12 se puede observar los costes asociados a recursos humanos que han sido necesarios para la realización del trabajo. El precio por hora es el coste calculado en el apartado 7.1 y el número de horas es una estimación calculada

a partir del punto 1.4 del documento, donde se realizó la planificación de este proyecto.

### 7.3 Coste total de proyecto.

La siguiente tabla se muestra todos los costes que repercuten en el desarrollo inicial del presente trabajo. Los costes asociados a servicios se han calculado como costes anuales, aunque, por ejemplo, en el caso de lightsail, el pago se realiza de forma mensual. Estos servicios pueden considerarse también costes de mantenimiento del sistema.

| Concepto | Coste |
|---|---|
| Coste prototipos | 519,40 € |
| Coste mano de obra ingeniería | 7640,00 € |
| **Servicios (Suscripción anual)** | |
| Licencia Sigfox 1 dispositivo [76] | 16,13 € |
| Servidor Lightsail [77] | 38,50 € |
| **Total** | **8214,03 €** |

*Tabla 13: Coste total de trabajo*

# 8. Conclusiones

En este trabajo se ha propuesto, desarrollado y analizado un nuevo modelo de arquitectura formado por una combinación de diferentes tipos de comunicaciones inalámbricas como son las tecnologías WPAN y LPWAN. Esta arquitectura tiene como objetivo la detección de diferentes tipos de emergencias, tanto de forma automática como habilitado externamente por el usuario.

El sistema de emergencias resultante está formado no solo por un conjunto de sensores, sino que también incluye el desarrollo de un servidor donde se almacena dicha información y mediante un cliente web, el usuario puede consultar y controlar todas las alarmas que se producen en tiempo real, así como editar las bases de datos con la información relevante del proyecto.

Tras el desarrollo del trabajo se puede concluir que gran parte de los objetivos propuestos se han conseguido. Estos objetivos inicialmente eran:

- **Estudio, análisis y comparativa de las tecnologías relacionadas con el proyecto, tanto en el ámbito de detección de emergencias como en comunicaciones inalámbricas.** En este caso, tanto el capítulo relacionado con el estado del arte como el relacionado con el desarrollo de la arquitectura se profundiza en ambos campos dando al lector una visión global de las tecnologías más actuales en los campos propuestos, como en las tecnologías directamente relacionadas con el desarrollo del trabajo.

- **Desarrollo de una solución completa de un sistema de emergencias basado en comunicaciones inalámbricas con las siguientes propiedades:**

    - **Robustez y fiabilidad**
    - **Gran cobertura**
    - **muy bajo Consumo**
    - **Coste reducido**

    Tras todas las pruebas realizadas sobre el sistema, podemos considerar que se trata de un sistema **fiable,** eléctricamente y en lo que a lectura de temperatura se refiere. En cambio, a la hora de realizar lecturas sobre la concentración de gases, la dispersión de valores entre los nodos es demasiado grande como para considerarlo un sistema totalmente funcional en ese aspecto.
    Respecto a la **cobertura** del sistema, se puede decir que el sistema dispone de un amplio alcance, ya que tanto la cobertura entre los gateway de cada vivienda como del gateway que da enlace a la nube se realiza a través comunicaciones LPWAN dotándole de grandes ventajas en este ámbito.
    El **coste** del sistema, que se puede observar en la valoración económica, no es tan bajo como inicialmente se tenía previsto, esto es debido a que el coste de los componentes, fabricación de PCB y montaje tienen un

gravamen por haber realizado un pedido orientado a prototipado. De haber realizado un pedido con un volumen de producción los costes se reducirían en gran medida.
Respecto al **consumo**, los sistemas que funcionan como gateway requieren inicialmente de conexión a la red eléctrica, aunque disponen de batería para que en caso de corte eléctrico puedan seguir trabajando correctamente durante horas. El mayor problema se ha detectado en los dispositivos sensores, los cuales, se ha concluido que el consumo es demasiado elevado y por lo tanto y a pesar de funcionar la mayor parte del tiempo en un modo de ahorro energético, la autonomía queda reducida a unos pocos meses, cuando este tipo de sistemas deberían ser recargados después de años en funcionamiento.

A pesar de haber alcanzado gran parte de los objetivos propuestos en este trabajo, queda mucho camino por recorrer. Por ello es necesaria la propuesta de una serie de mejoras que reduzcan o eliminen las debilidades encontradas.
A continuación, se proponen algunas:

- **BLE Mesh.** La topología de la red WPAN desarrollada en el trabajo, basada en tecnología bluetooth low energy finalmente fue en una arquitectura de estrella. Esto fue debido principalmente a que existen multitud de ejemplos a la hora de desarrollar el firmware para los dispositivos. Aun así, la topología óptima para este tipo de sistemas seria topología de malla o *Mesh*. Esta nueva arquitectura de red de los dispositivos permitiría un aumento de la cobertura y aportaría principalmente que todos los dispositivos participantes pudieran conectarse unos a otros sin necesidad de estar al alcance del cliente o provisionador, lo que podría darle acceso al destino sin tenerlo visible.

- **Medida de parámetros.** Respecto a la medida de parámetros, debido a la complejidad del sensor BME680, no se ha podido realizar un profundo estudio de su funcionamiento, sobre todo en el aspecto de la detección de gases. Durante el desarrollo del trabajo se partió de un ejemplo en Arduino **[78]** para la cuantificación de la medida de concentración de gases a través del sensor. Los valores leídos en diferentes nodos diferían en gran medida, a pesar de que demostraron en todos ellos su sensibilidad ante la exposición de los elementos. Una alternativa al uso de librerías desarrolladas por terceras partes es el uso de paquetes binarios desarrollado por el propio fabricante **[79]**. Bosch proporciona una serie de elementos binarios incrustables para el cálculo del parámetro IAQ. El problema de estos elementos es que es un código cerrado y la integración de este en cualquier lenguaje de programación resulta un poco complicada. A pesar de estas dificultades sería conveniente un estudio más profundo de esta opción para poder obtener una lectura de parámetros más fiable y con menos dispersión en lo referente a la medida de gases.

- **WiFi. Configuración inicial. Actualización de firmware.** Durante el desarrollo del trabajo, en los apartados de configuración de los diferentes dispositivos que componen el sistema de comunicación, se ha podido

observar que existen diferentes parámetros configurables que indican a que subred pertenece cada dispositivo. Por ejemplo, para que un nodo pueda conectarse a un gateway es necesario que sus identificadores UUID de servicio sean idénticos. También el gateway de la vivienda deberá aportar un numero identificador que lo asocie a la propia dirección de la vivienda. Todos estos datos se configuran de forma manual cuando se realiza la compilación y posterior descarga del dispositivo, por lo tanto, a la hora de configurar la red es necesario programar cada dispositivo de forma personalizada a través del puerto de descarga. Para evitar esto, la opción más apropiada es el uso de la conectividad WiFi para la configuración de estos parámetros.

El proceso de configuración de cada dispositivo sería el siguiente:

- Lectura de memoria por parte de cada dispositivo. Si este no detecta ningún identificador de servicio, red o gateway. Habilitará el transceptor WiFi en modo punto de acceso (*Access Point Station*) donde estará alojada una página web de configuración del dispositivo.

- El personal encargado de la instalación podrá acceder desde cualquier terminal con conectividad WiFi, como un teléfono móvil, tablet u ordenador portátil direccionando el cliente web con la dirección IP del dispositivo y una conexión establecida punto a punto con el dispositivo. En ella, podrá realizar todas las configuraciones necesarias sobre el dispositivo.

- Tras finalizar la configuración y guardarla, el dispositivo se reiniciará. Al reiniciar, volverá a leer la información almacenada en su memoria. Como en este caso es posible la inicialización de todos los parámetros de identificación de red, no será necesario volver a encender el transceptor WiFi y podrá realizar su funcionamiento normalmente.

Otra posible mejora que puede introducir la conectividad WiFi es su uso para realizar la actualización de firmware. Esta actualización podría realizarse "*Over The Air*" u OTA, es decir, realizar la descarga del fichero de actualización cuando el dispositivo está en funcionamiento y actualizar tras un reinicio. Para realizar este tipo de descargas sería necesario su señalización o notificación a través de una orden externa como la pulsación de un botón, o por una orden emitida desde el cliente al otro lado de la red, gracias a una posible comunicación bidireccional en la arquitectura.

- **Uso de LoRaWAN como tecnología final a la nube.**
  El uso de Sigfox como sistema de comunicación final para tecnología de comunicación final de las alarmas generadas viene dado por la poca cobertura del sistema LoRaWAN en España actualmente. Esto por lo tanto conlleva una serie de costes asociados a la cuota de servicio de

Sigfox. En un futuro próximo este problema por parte de la tecnología LoRa podría desaparecer y aportar una nueva visión al proyecto, ya que se trata de una tecnología con similares características en lo que alcance se refiere, pero con la ventaja de disponer de un mayor ancho de banda, por ejemplo, se podría enviar cada cierto tiempo parámetros medidos para realizar estudios de *Bigdata* o usar LoRa en topología *Mesh* para aumentar la cobertura del sistema.

- **Arquitectura con comunicación bidireccional**. En el desarrollo de la arquitectura, la comunicación implementada solamente se realiza en un sentido, desde el nodo, hasta su visualización por parte del usuario. En cambio, el usuario no dispone de ninguna herramienta para el envío de ordenes al sistema. Como se pudo observar en el apartado 5.3.4 existen métodos de comunicación que permiten la comunicación bidireccional a través de Sigfox, como es el uso de las librerías API REST proporcionadas por el fabricante, esto permitiría realizar envíos que podrían:

    - Enviar una orden a todos los dispositivos para comprobar el estado del repositorio de actualizaciones firmware, de tal manera que, si existe una versión nueva, sabría exactamente en qué momento debe activar su transceptor WiFi y poder comprobarlo, ahorrando posibles habilitaciones de la comunicación programadas.

    - Cambios en la configuración de los parámetros umbrales de detección, por ejemplo, subir la temperatura limite, según la estación del año.

    - Petición de lectura de parámetros para realizar estudios de *machine learning*. Aunque su ancho de banda sea muy limitado.

- **Modificaciones en placa para uso de pilas de litio**. En el diseño actual no se tuvo en cuenta el modo Deep sleep y por lo tanto no se incluyeron las configuraciones hardware necesarias para su funcionamiento, haciendo necesario realizar modificaciones. Tampoco se incluyó un encapsulado para el posicionamiento de una pila de litio en caso del uso de este tipo de alimentación, ni un jumper para seleccionar un tipo de alimentación u otra en caso de elegir este tipo de fuente de energía.
  También son necesarias otras modificaciones, como el cambio del GPIO encargado de la detección de pulsación de emergencia que está en conflicto directo con el ADC1 del microcontrolador.

- **Energy Harvesting en dispositivos de alto consumo como los gateway.** El alto consumo medido en los dispositivos que funcionan como puertas de enlace obliga a su conexión a la red eléctrica mediante conversores AC/DC. Esto podría evitarse mediante el diseño de sistemas *energy harvesting* que permitirían la carga de la batería de litio y por lo tanto más autonomía al sistema, incluso haciendo innecesario el cableado a la red.

- **Cambio de microcontrolador:** Tras los cálculos de estimación de consumo y posterior medición del consumo sobre el sistema real, se llegó a la conclusión de que la unidad de procesamiento consumía más de los límites establecidos para la función de nodo. Por ello es necesaria la selección de una unidad de procesamiento con una eficiencia energética mayor que permita un aumento de la autonomía.

- **Mecánica:** Los dispositivos diseñados no disponen de carcasa mecánica. Para proporcionar un producto final es necesario el diseño de una caja que permita la protección y ocultación de todos los componentes del dispositivo.

# 9. Glosario

**ADC:** Analogic digital converter
**AJAX:** Asynchronous JavaScript And XML
**API:** application programming interface
**AWS:** Amazon Web Services.
**BBDD**: Bases de datos
**BGA**: Ball grid array
**BLE:** Bluetooth Low Energy
**BOM:** Bill of materials
**BT**: Bluetooth
**CAD:** computer-aided design
**CGNAT**: Carrier Grade NAT
**CSS:** Chirp spread spectrum
**ENIG:** Electroless Nickel/ Immersion Gold
**FR4**: Flame Retardant 4
**GAP:** Generic Access Profile
**GATT:** Generic Attribute Profile
**GPIO**: General purpose input/output
**HASL:** hot air solder leveling
**HTML:** HyperText Markup Language
**HTTP**: Hypertext Transfer Protocol
**I2C:** Inter-Integrated Circuit
**IAg**: Immersion Silver Plating
**IAQ**: Indoor Air Quality
**IEEE**: Institute of Electrical and Electronics Engineers
**IoT:** internet of things
**IP:** Internet Protocol
**IR:** Infrared
**ISM:** Industrial, Scientific and Medical frequency band
**LAN:** Local Area Network
**LDO:** Low Drop Out
**LiPo:** Litio polimerer
**LPWAN:** Low Power Wide Area Network
**LoRaWAN:** LoRa World Area Network
**NAT**: Network Address Translation
**NB-IoT**: Narrow Band- Internet of Things
**OTA**: Over the Air
**PCB:** Printed Circuit Board
**PWM**: pulse-width modulation
**RAM**: Random Access Memory
**ROHS:** Restriction of Hazardous Substances
**ROM**: Read Only Memory
**RF:** Radio Frequency
**SoM:** System on Module
**SPI**: Serial Peripheral Interface
**SPST:** Single Pole Single Throw
**UNB:** Ultra Narrow Band
**USB:** Universal serial bus
**UUID**: Universally unique identifier

**VPS**: Virtual Private Service
**WLAN:** Wireless Local Area Network
**WPAN**: Wireless Personal Area Network
**WAN:** Wide Area Network
**WSN:** Wireless Sensor Network
**DMIPS**: Dhrystone Million Instructions Per Second

# 10. Bibliografía

**[1]** Autor: Cofem. ***Detectores de incendios, humo, gases domésticos.*** Visitado el 22/02/2020
http://www.cofem.com/productos/deteccion-domestica/detector-domestico-de-humos/

Autor: Enmicasalomejor***. Comparativa de mejores detectores de humo domésticos.*** Visitado el 29/02/2020.
https://enmicasalomejor.com/mejor-detector-de-humo-domestico/

**[2]** Autor: Securitas Direct. ***Productos detección de incendios de Securitas Direct.*** Visitada el 29/02/2020
https://www.securitasdirect.es/es/servicios/alarma-contra-incendios

**[3]** Autor: The fire Brigades Union. ***Why emergency response times matter to firefighters and the public. The Fire Brigades Union***. Visitado 27/02/2020
https://www.epsu.org/sites/default/files/article/files/6367-Its-about-time-LOW-RES2.pdf

**[4]** Autor: Fundación Mapfre. ***"Víctimas de incendios en España en 2018. © Fundación MAPFRE y APTB, 2019".*** *Visitado* el 29/02/2020
https://noticias.fundacionmapfre.org/wp-content/uploads/2019/10/Estudio_Victimás_Incendios_Espana_2018.pdf

**[5]** Autor: Adecco Institute. ***10 beneficios del IoT para empresas.*** Visitado el 21/02/2020
https://www.adeccoinstitute.es/futuro-del-trabajo-y-tecnologia/10-beneficios-del-iot-para-empresas/

**[6]** Autor: Universidad de les Illes Balears***. Prevención de riesgos laborales. Universidad de Islas Baleares.*** Visitado el 4/03/2020
https://www.uib.cat/depart/dqu/dquo/dquo2/MásterSL/ASIG/PDF.old/242SIS~1.PDF

**[7]** Autor: Joan Pau Esplugas Vidal. ***Guía para el diseño, uso y mantenimiento de los sistemas de detección automática de incendios. Prevención de riesgos laborales. ASEPEYO.***
© ASEPEYO Mutua Colaboradora con la Seguridad Social nº 151 1ª Edición, junio 2016

**[8]** Autor: Oversea. ***Extintores oversea.*** Visitado el 05/03/2020
http://oversea.cl/detdetectores.html

**[9]** Autor: TecnoSeguro. ***Revista digital especializado en seguridad electrónica.*** Visitado el 07/03/2020
https://www.tecnoseguro.com/tutoriales/pro/cual-es-la-diferencia-entre-un-detector-de-humo-ionico-y-uno-fotoelectrico

**[10]** Autor: José Luis Villanueva Muñoz. ***NTP 40: Detección de incendios. Guía de buenas prácticas. Ministerio de trabajo y asuntos sociales de España Año 1983.*** Visitado el 06/03/2020
https://www.insst.es/documents/94886/326853/ntp_040.pdf/9ba14077-7121-415a-a6e9-d8c9f2a4031c

**[11]** Autor: Emilio Turmo Sierra. ***NTP-215: Detectores de humos. Guía de buenas prácticas. Ministerio de trabajo y asuntos sociales de España*** **Año 1988.** Visitado el 06/03/2020
https://www.insst.es/documents/94886/327166/ntp_215.pdf/78d02563-f887-4ed3-a3cf-371c1a35c0f5

**[12]** Autor: Emilio Turmo Sierra. ***NTP 185: Detección automática de incendios. Detectores térmicos. Guía de buenas prácticas. Ministerio de trabajo y asuntos sociales de España***. **Año 1988** *Visitado el 06/03/2020*
https://www.insst.es/documents/94886/326801/ntp_185.pdf/e9fe7b54-3c3d-4930-bf45-905f0e38d84f

**[13]** Autor: 802.15 WG. ***Standard IEEE 802.15.1.*** Visitado el 16/03/2020
https://standards.ieee.org/standard/802_15_1-2005.html

**[14]** Autor: Bluetooth SIG. ***Bluetooth 4.2 Specification.*** Visitado el 16/03/2020
https://www.bluetooth.org/docman/handlers/downloaddoc.ashx?doc_id=441541

**[15]** Autor: Bluetooth SIG. ***Bluetooth 5.0 Specification.*** Visitado el 16/03/2020
https://www.bluetooth.org/docman/handlers/DownloadDoc.ashx?doc_id=421043

**[16]** Autor: 802.15 WG. ***Standard IEEE 802.15.4.*** Visitado el 16/03/2020
https://standards.ieee.org/standard/802_15_4-2015.html

**[17]** Autor: Sparkfun. ***IR Communication. Tutorial.*** Visitado el 07/05/2020
https://learn.sparkfun.com/tutorials/ir-communication/all

**[18]** Autor: Wikipedia. ***Narrowband Internet of things. Wikipedia.*** Visitado el 08/05/2020
https://en.wikipedia.org/wiki/Narrowband_IoT

**[19]** ***LoRa Alliance.*** Visitado el 16/03/2020
https://lora-alliance.org/

**[20]** ***Sigfox.*** Visitado el 16/03/2020
https://www.sigfox.com/en

**[21]** Marcio Soares Ferreira, Ricardo Augusto R. Oliveira, Jorge Sa Silva. ***Low-Energy Smart Cities network with LoRa and Bluetooth.***
Celio 2019 7th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud). 2019

**[22]** Autor: Semtech. ***Application Brief. Fire Detection.*** Visitado el 09/03/2020 https://www.semtech.com/uploads/technology/LoRa/app-briefs/Semtech_HB_FireDetection_AppBrief-FINAL.pdf

**[23]** ***Siterwell, Smart Security Ecosystem Leader*.** Visitado el 14/03/2020 https://www.china-siter.com/siterwell.html

**[24]** Karwan MUHEDEN, Ebubekir ERDEM, Sercan VANÇIN. CINTI. ***Design and Implementation of the Mobile Fire Alarm System Using Wireless Sensor Networks*** 2016 • 17th IEEE International Symposium on Computa tional Intelligence and Informatics • 17–19 November, 2016 • Budapest, Hungary

**[25]** ***HEYI Alarm Systems,*** Visitado el 15/03/2020 https://www.heyitech.com/alarm-system/ http://www.globalmarket.com/product-info/cctv-security-systems-monitoring-alarm-installers-11211668.html

**[26]** ***Securitas Direct. Sistema de alarmas para el hogar****.* Visitado el 15/03/2020 https://www.securitasdirect.es/es/servicios/alarma-contra-incendios

**[27]** Daniel Adorno, Salviano Soaresy, Jose Limaz, and Antonio Valente. ***Evaluation of LP-WAN Technologies for Fire Forest Detection Systems*** *ALLSENSORS 2019: The Fourth International Conference on Advances in Sensors, Actuators, Metering and Sensing. Fecha de publicación: 2019*

**[28]** ***Mapa de cobertura Sigfox*.** Visitado el 04/06/2020 https://www.sigfox.com/en/coverage

**[29]** ***Different Wi-Fi Protocols and Data Rates.*** Visitado el 15/03/2020

https://www.intel.com/content/www/us/en/support/articles/000005725/network-and-i-o/wireless-networking.html

**[30]** ***Iot Standards and Protocols.*** Visitado el 15/03/2020 https://www.postscapes.com/internet-of-things-protocols/

**[31]** Mahmoud Elkhodr, Seyed Shahrestani and Hon Cheung. ***Emerging Wireless Technologies in the internet of things: A comparative Study.*** International Journal of Wireless & Mobile Networks (IJWMN) Vol. 8, No. 5, Fecha de publicación: octubre 2016

**[32]** Kais Mekki, Eddy Bajic, Frederic Chaxel, Fernand Meyer. ***A comparative study of LPWAN technologies for large-scale IoT deployment***, Fecha de publicación: March 2019

**[33]** Ian F. Akyildiz, Weilian Su, Yogesh Sankarasubramaniam, and Erdal Cayirci. Georgia Institute of Technology, ***A Survey on Sensor Networks.*** Fecha publicación: 2002

**[34]** Tatiana C. Muñoz H., Ing* Demetrio A. Ovalle C., PhD* Alcides J. Montoya C., MSc. ***Hacia la evaluación del desempeño de un modelo de integración de agentes inteligentes & WSN a través de métricas*****.** REVISTA COLOMBIANA DE COMPUTACIÓN. Fecha publicación: 05/03/2010

**[35]** Gongsheng Huang, Pei Zhou, Linfeng Zhang
***Wireless sensor network for HVAC applications: a review***
12th International Conference on Sustainable Energy technologies. Hong Kong 26-29th August, 2013

**[36]** ***Familia de microcontroladores STM32WB****.* Visitado el 08/05/2020
https://www.st.com/en/microcontrollers-microprocessors/stm32wb-series.html

**[37]** Autor: NORDIC. ***Microcontrolador Nordic nRF52832.*** Visitado el 08/05/2020
https://www.nordicsemi.com/-/media/Software-and-other-downloads/Product-Briefs/nRF52-Series-SoC-PB-50.pdf?la=en&hash=47F6BE0898D610DCB171CEDAE96977CBEC1CBBBA

**[38]** ***Familia espressif ESP32.*** Visitado el 08/05/2020
https://www.espressif.com/en/products/socs/esp32/overview

**[39]** Autor: Espressif. ***Datasheet Espressif ESP32-PICO-D4***. Visitado el 25/04/2020
https://www.espressif.com/sites/default/files/documentation/esp32-pico-d4_datasheet_en.pdf

**[40]** Autor: Murata. ***LoRa Module muRata CMWX1ZZABZ-078.*** Visitado el 08/05/2020
https://wireless.murata.com/type-abz-078.html

**[41]** Autor: Laird Connect. ***Laird módulo LoRa y BLE*****.** Visitado el 08/05/2020
https://www.lairdconnect.com/wireless-modules/lorawan-solutions/sentrius-rm1xx-lora-ble-module

**[42]** Autor: Semtech***. Familia de transceptores de LPWAN Semtech SX1276/77/78/79.*** Visitado el 25/04/2020
https://www.semtech.com/products/wireless-rf/lora-transceivers/sx1276#download-resources

**[43]** Autor: AM5***. Sensor de calidad del aire CCS811.*** Visitado el 08/05/2020
https://cdn.sparkfun.com/assets/learn_tutorials/1/4/3/CCS811_Datasheet-DS000459.pdf

**[44]** Autor: Texas Instrument***. Sensor de temperatura y humedad HDC1080.*** Visitado el 08/05/2020
http://www.ti.com/lit/ds/symlink/hdc1080.pdf?&ts=1588947470484

**[45]** Autor: Bosch***. Sensor ambiental Bosch BME680.*** Visitado el 25/04/2020 https://www.bosch-sensortec.com/products/environmental-sensors/gas-sensors-bme680/

**[46]** Autor: CUI***. E-SWITCH Buzzer 2KHz de fabricante CUI.*** Visitado el 25/04/2020.
https://www.cuidevices.com/product/resource/cmt-1285c-035.pdf

**[47]** Autor: E-SWITCH***. Pulsador SPST E-SWITCH.*** Visitado el 08/05/2020 https://www.mouser.es/datasheet/2/140/P010454B-267753.pdf

**[48]** Autor: Diodes. ***Regulador lineal LDO AP2112K****.* Visitado el 26/04/2020 https://www.diodes.com/assets/Datasheets/AP2112.pdf

**[49]** Autor: Microchip. ***Controlador de carga de baterías de litio MCP73831/2.*** Visitado el 26/04/2020
http://ww1.microchip.com/downloads/en/DeviceDoc/20001984g.pdf

**[50]** ***Silicon Labs. CP2104. Conversor USB a UART.*** Visitado el 26/04/2020 https://www.silabs.com/documents/public/data-sheets/cp2104.pdf

**[51]** ***Herramienta de diseño esquemático y PCB libre KiCAD.*** Visitado el 26/04/2020 https://www.kicad-pcb.org/

**[52]** Autor: multi-cb. ***Recomendaciones sobre simetría en placas de circuito impreso.*** Visitado el 26/04/2020 https://www.multi-circuit-boards.eu/en/pcb-design-aid/copper-balance.html

**[53]** Autor: Al Wright. ***Stackup, configuraciones de PCB multicapa y dieléctricos.*** Visitado el 26/04/2020 https://blog.epectec.com/balancing-layers-in-your-pcb-layout

**[54]** Autor: Intel. ***PCB Stackup Design Considerations for Intel® FPGAs****.* Visitado el 26/04/2020
https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/an/an613.pdf

**[55]** Autor: JLCPCB. ***Stackup JLCPCB 4 capas y 1,6 mm de grosor.*** Visitado el 05/05/2020.
https://cart.jlcpcb.com/impedance?_ga=2.107828609.2058937075.1588686377-763207696.1584001333&_gac=1.192168408.1586190615.CjwKCAjwpqv0BRABEiwA-TySwZun_vENm9XfGbHF-GfDVZ-luCI0A-BbEuVvCwy5zFICTHPKr2mCKxoCdIoQAvD_BwE

**[56]** ***Herramienta para diseño electrónico Saturn PCB Tool kit.*** Visitado el 26/04/2020 http://saturnpcb.com/pcb_toolkit/

**[57]** ***Aliexpress.*** Visitado el 04/06/2020
https://es.aliexpress.com/

**[58]** ***Lilygo T5. Placa comercial basada en microcontrolador ESP32.*** Visitado el 05/05/2020
http://www.lilygo.cn/prod_view.aspx?TypeId=50031&Id=1169
**[59]** ***Lilygo TTgo ESP32 LoRa V2,4.*** Visitado el 05/052020
http://www.lilygo.cn/prod_view.aspx?TypeId=50003&Id=1271

**[60]** ***Pycom Lopy4. Modulo basado en ESP32 con cuádruple conectividad.*** Visitado el 05/05/2020
https://pycom.io/product/lopy4/

**[61]** ***Pycom Pymakr V3 placa de expansión.*** Visitado el 05/05/2020.
https://pycom.io/product/expansion-board-3-0/

**[62]** Autor: Bluetooth SIG. ***Lista oficial de perfiles BLE.*** Visitado el 26/04/2020.
https://en.wikipedia.org/wiki/List_of_Bluetooth_profiles

**[63]** Autor: Bluetooth SIG. ***Lista oficial de servicios BLE.*** Visitado el 26/04/2020 https://www.bluetooth.com/specifications/gatt/services/

**[64]** Autor: Bluetooth SIG. ***Lista oficial de características BLE.*** Visitado el 26/04/2020
https://www.bluetooth.com/specifications/gatt/characteristics/

**[65]** Autor: Neil Kolban. ***Repositorio de Librería BLE Arduino para ESP32.*** Visitado el 05/05/2020
https://github.com/nkolban/ESP32_BLE_Arduino

**[66]** Autor: Sigfox. ***Opciones de gestión de información almacenada en backend de Sigfox***. Visitado el 08/05/2020
https://build.sigfox.com/backend-callbacks-and-api

**[67]** Autor: Sigfox. ***Términos y condiciones en el uso de API y funciones de Sigfox.*** visitado el 08/05/2020
https://storage.sbg.cloud.ovh.net/v1/AUTH_669d7dfced0b44518cb186841d7cbd75/prod_docs/55745493-Sigfox_API-Portal_CTU.pdf

**[68]** ***AWS Amazon Web Services.*** Visitado el 12/05/2020
https://aws.amazon.com/es/?nc2=h_lg

**[69]** ***HTML, lenguaje de marcas de hipertexto.*** Visitado el 13/05/2020
https://developer.mozilla.org/es/docs/Web/HTML
https://es.wikipedia.org/wiki/HTML

**[70]** ***JavaScript. Lenguaje de programación.*** Visitado el 13/05/2020
https://developer.mozilla.org/es/docs/Web/JavaScript

**[71]** ***jQuery y Ajax: Desarrollo de peticiones a servidores web.*** Visitado el 13/05/2020
https://jquery.com/
https://es.wikipedia.org/wiki/AJAX
https://api.jquery.com/jquery.ajax/

**[72]** ***Código referencia para medición de calidad del aire a partir de sensor BME680***. Visitado el 08/05/2020
https://github.com/G6EJD/BME680-Example

**[73]** ***Acetona.*** Visitado el 15/05/2020
https://es.wikipedia.org/wiki/Acetona

**[74]** ***Mouser. Distribuidor de componentes electrónicos.*** Visitado el 23/05/2020
https://www.mouser.es/

**[75]** Autor: CC. OO, UGT, FESMC y TECNIBERIA. ***Convenio Colectivo del sector de empresas de ingeniería y oficinas de estudios técnicos***. Visitado el 23/05/2020
https://www.boe.es/boe/dias/2019/10/18/pdfs/BOE-A-2019-14977.pdf

**[76]** ***Costes de servicio Sigfox***. Visitado el 23/05/2020
https://buy.sigfox.com/buy/offers/ES

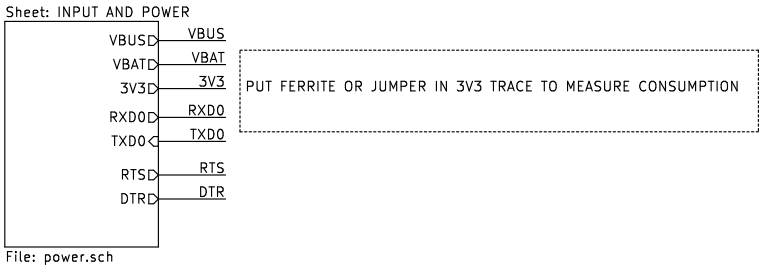
**[77]** ***Tabla de precios de servicios Lightsail de Amazon.*** Visitado el 23/05/2020
https://aws.amazon.com/es/lightsail/pricing/

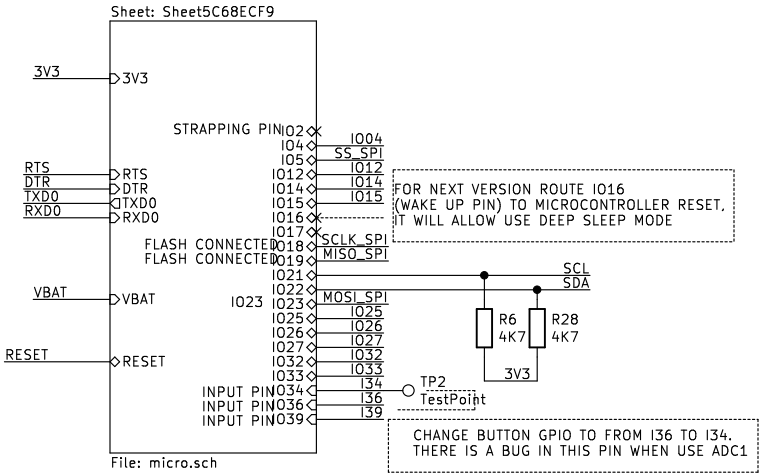
**[78]** Autor: G6EJD. ***Ejemplo algoritmo basado en IAQ para BME680.*** Visitado el 24/05/2020
https://github.com/G6EJD/BME680-Example

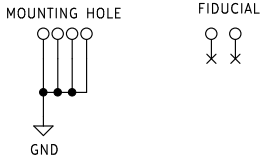
**[79]** ***Bosch Sensortec Envionmental Cluster(BSEC).*** Visitado el 24/05/2020
https://www.bosch-sensortec.com/software-tools/software/bsec/

# 11. Anexos

# ANEXO A: DOCUMENTACION HARDWARE

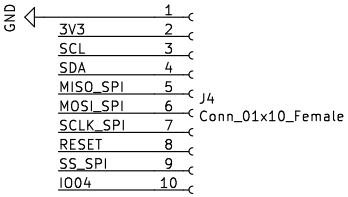


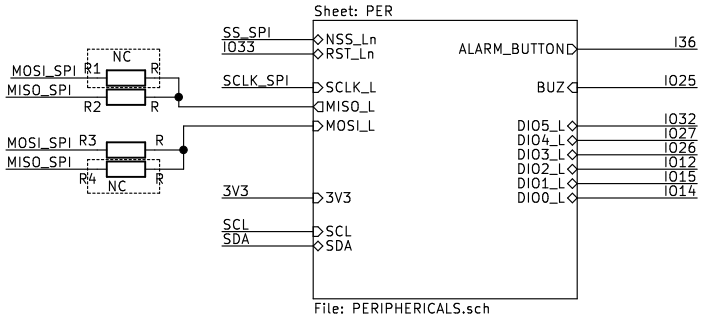



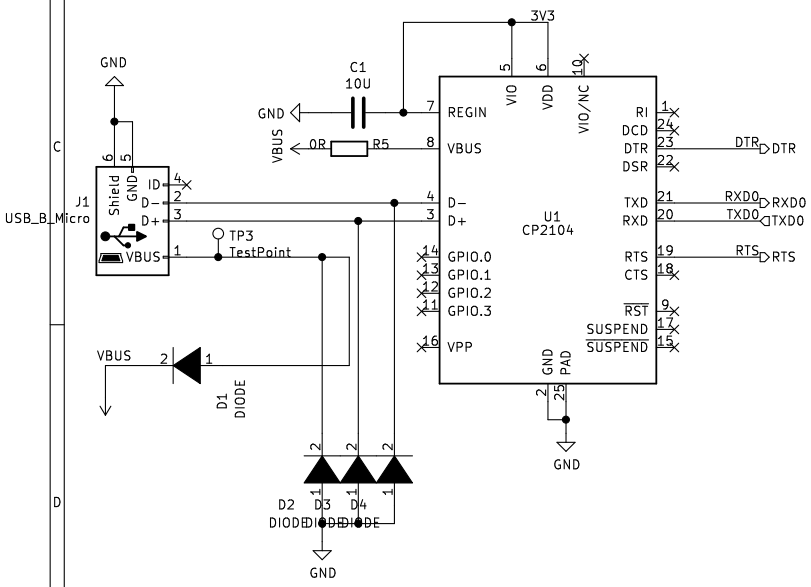


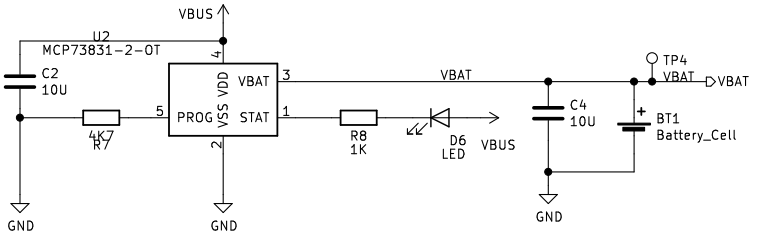


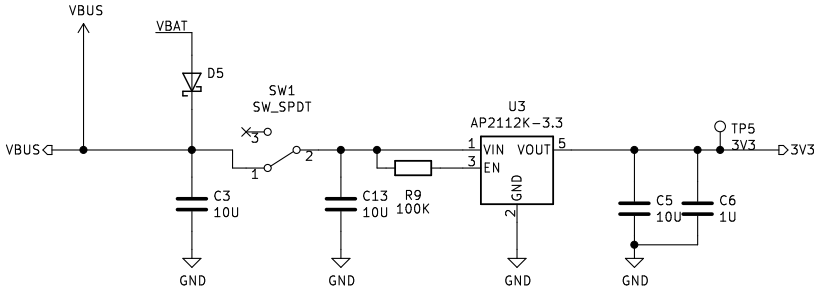


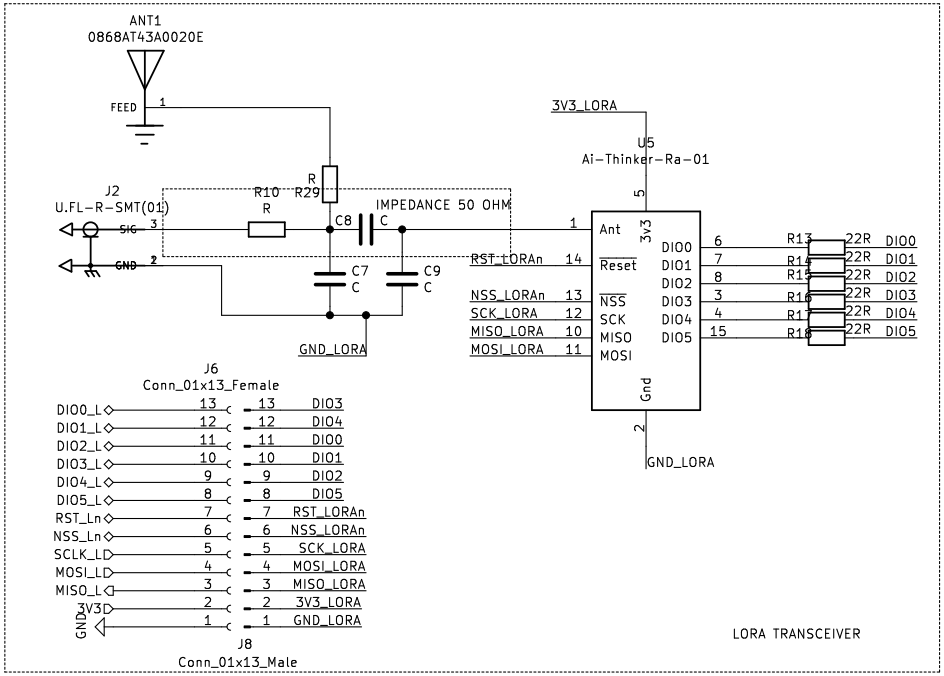

| | | |
|---|---|---|
| Sheet: /<br>File: home.sch | | |
| **Title:** | | |
| Size: A | Date: | **Rev:** |
| KiCad E.D.A. kicad (5.1.2)-1 | | Id: 1/4 |

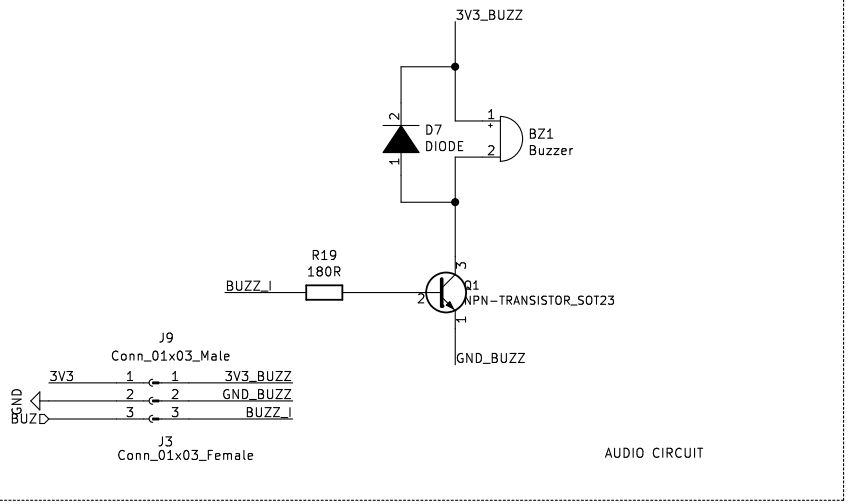


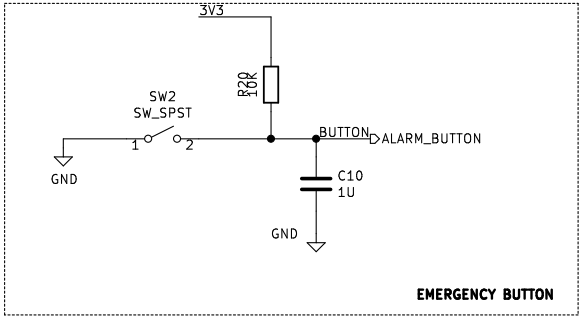



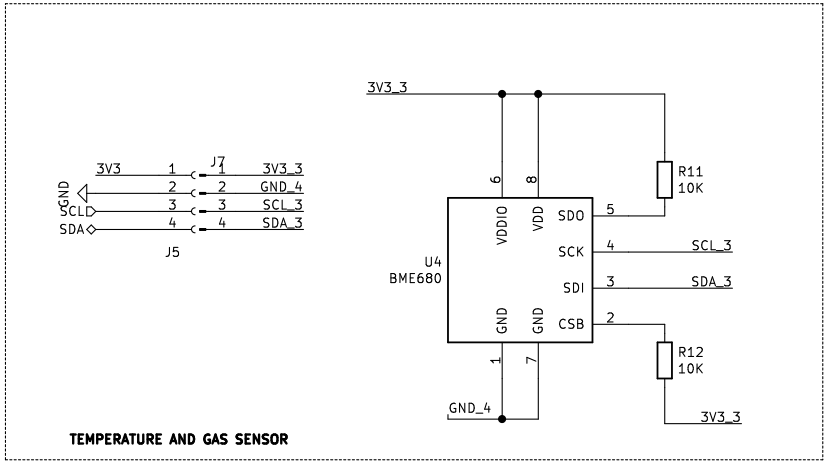




| | | |
|---|---|---|
| Sheet: /INPUT AND POWER/ | | |
| File: power.sch | | |
| **Title:** | | |
| Size: A4 | Date: | **Rev:** |
| KiCad E.D.A. kicad (5.1.2)-1 | | Id: 2/4 |

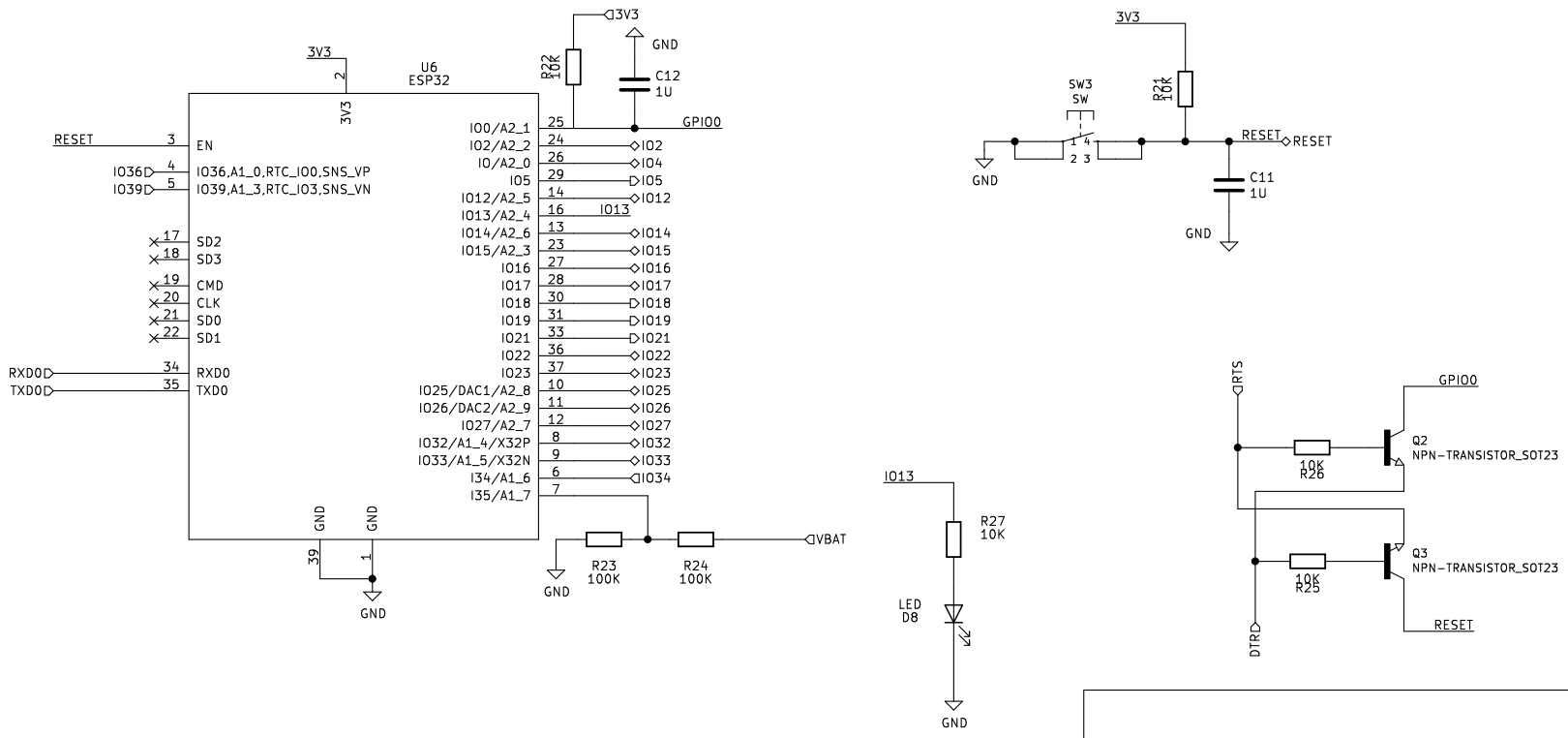



| | | |
|---|---|---|
| Sheet: /PER/ | | |
| File: PERIPHERICALS.sch | | |
| **Title:** | | |
| Size: A4 | Date: | **Rev:** |
| KiCad E.D.A. kicad (5.1.2)-1 | | Id: 3/4 |

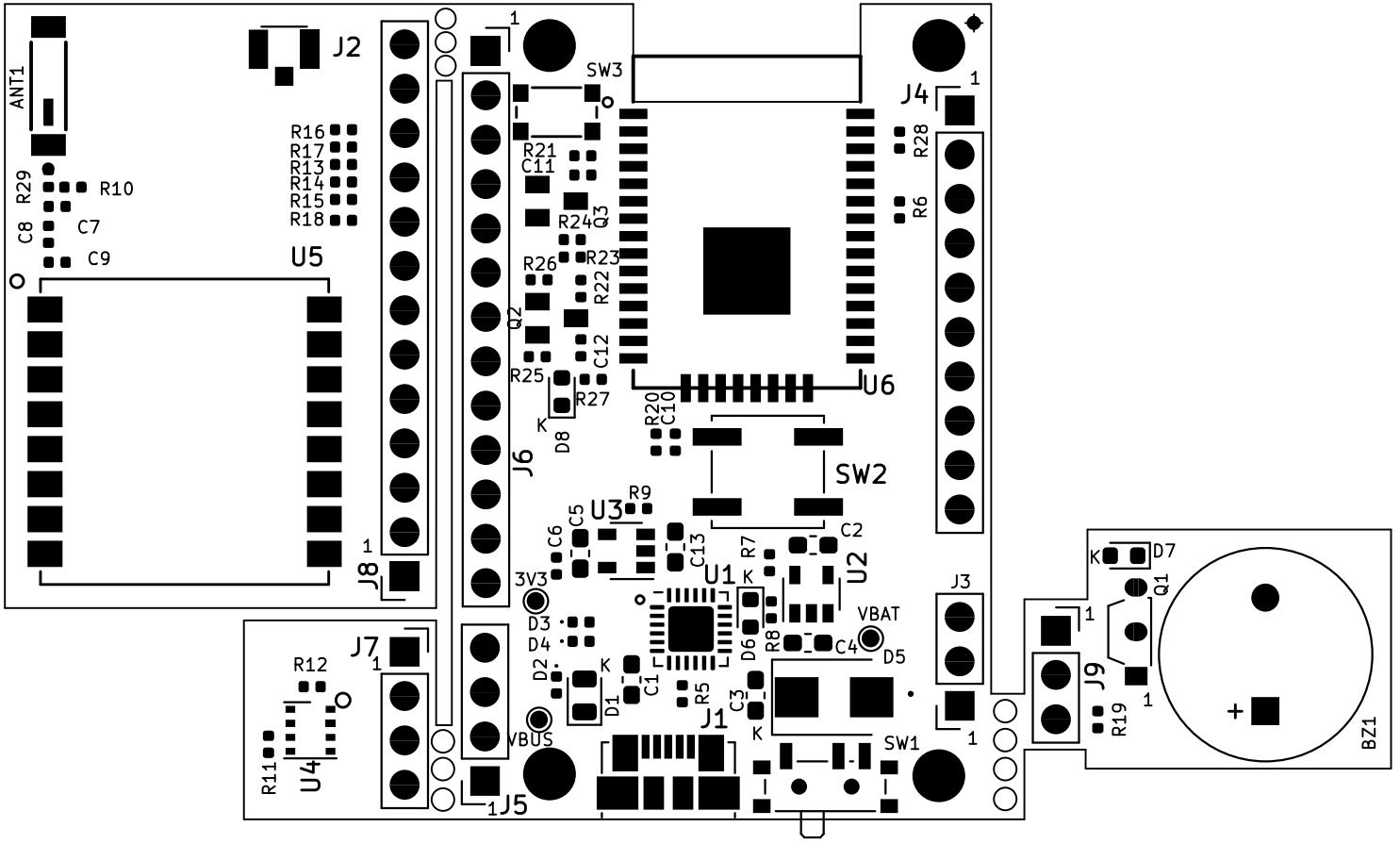



Sheet: /Sheet5C68ECF9/
File: micro.sch

**Title:**

| Size: A4 | Date: | **Rev:** |
|---|---|---|
| KiCad E.D.A. kicad (5.1.2)-1 | | Id: 4/4 |

| Manufacturer | Manufacturer Part Number | Designator | Qty | Description | Case | SMD ,BGA,LLC (QFN),OR TH | Top/Bottom | Polarity | Comment |
|---|---|---|---|---|---|---|---|---|---|
| FAIRCHILD | 2N5550TAR | Q1 | 1 | Medium power transistor 50V 0.5A | TO-92-3 Kinkeo | TH | T | Y | https://www.mouser.es/datasheet/2/308/2N5550-D-1801436.pdf |
| CUI | CMT-1285C-035 | BZ1 | 1 | Magnetic buzzer transducer | D:12 6,5mm | TH | T | Y | https://www.mouser.es/datasheet/2/670/cmt-1285c-035-1776800.pdf |
| BOURNS | CD0603-S01575 | D7 | 1 | DIODE 0603 General purpose 0,15A | 0603 | SMD | T | Y | https://www.mouser.es/datasheet/2/54/CD0603_1005-777241.pdf |
| SEMTECH | SX1276 MOD | U5 | 1 | LoRa Module SX1276 | nd | SMD | T | Y | https://es.aliexpress.com/item/32981592148.html?spm=a2g0s.9042311.0.0.673863c0dW8jwS |
| LILYGO | TTGO micro-32 | U6 | 1 | MICRO ESP32 module with ceramic antenna | nd | SMD | T | | https://es.aliexpress.com/item/32869180373.html?spm=a2g0s.9042311.0.0.673863c0dW8jwS |
| C&K | KMR421NGLFS | SW3 | 1 | Tactile siwtch 2N SPST | 5mmx2.6mm | SMD | T | | https://www.mouser.es/datasheet/2/60/kmr4-1382752.pdf |
| KINGBRIGHT | APG1608QBC/D | D8,D6 | 2 | ndard LEDs - SMD Blue, Water Clear 70mcd, 470 | 0603 | SMD | T | Y | https://www.mouser.es/datasheet/2/216/APG1608QBC-D-80547.pdf |
| TAIYO YUDEN | JMK105BJ105KP-F | C12,C11,C6,C10 | 4 | capacitor 0402 1uF | 0402 | SMD | T | | https://www.mouser.es/datasheet/2/396/mlcc02_e-1307760.pdf |
| VISHAY | CRCW0402100KFKEDC | R23,R24,R9 | 3 | resistor 0402 100K | 0402 | SMD | T | | https://www.mouser.es/datasheet/2/427/crcwce3-1762584.pdf |
| VISHAY | CRCW040210K0FKEDC | R25,R26,R21,R27,R22,R11,R12,R20 | 8 | resistor 0402 10K | 0402 | SMD | T | | https://www.mouser.es/datasheet/2/427/crcwce3-1762584.pdf |
| NEXPERIA | MMBT3904,215 | Q2,Q3 | 2 | NPN transistor SOT23 | SOT23-3 | SMD | T | | https://www.mouser.es/datasheet/2/916/MMBT3904-1599187.pdf |
| WURTH | 620102131822 | BT1 | 1 | Headers & Wire Housings WR-WTB 2mm MaleS | WR-WTB 2mm | SMD | B | | https://www.mouser.es/datasheet/2/445/620102131822-1718112.pdf |
| MURATA | GRM188R60J106ME84J | C1,C2,C3,C4,C5,C13 | 6 | capacitor 0603 10uF | 0603 | SMD | T | | https://www.mouser.es/datasheet/2/281/murata_03052018_GRM_Series_1-1310166.pdf |
| N SEMICONDUCTO | MBRS2040LT3G | D5 | 1 | diode schottky | SMB | SMD | T | Y | https://www.mouser.es/datasheet/2/308/MBRS2040LT3-D-1772065.pdf |
| AMPHENOL | 10118192-0001LF | J1 | 1 | micro usb connector | | SMD | T | | https://www.amphenol-icc.com/media/wysiwyg/files/documentation/datasheet/inputoutput/io_usb_micro.pdf |
| 3M | 929974-01-26-RK | J5 | 1 | pin socket 4 pin 2,54 mm female | 2,54mm | TH | T | | https://www.mouser.es/datasheet/2/1/ts0815-31700.pdf |
| 3M | 929974-01-26-RK | J6 | 1 | pin socket 13 pin 2,54 mm female | 2,54mm | TH | T | | https://www.mouser.es/datasheet/2/1/ts0815-31700.pdf |
| AMPHENOL | 93992-436HLF | J7 | 1 | pin socket 04 pin 2,54 mm male | 2,54mm | TH | T | | https://www.amphenol-icc.com/media/wysiwyg/files/documentation/datasheet/boardwiretoboard/bwb_bergstik.pdf |
| VISHAY | CRCW04020000Z0EDC | R2,R3,R5,R10,R29 | 5 | 0R 0402 resistor | 0402 | SMD | T/B | | https://www.mouser.es/datasheet/2/427/crcwce3-1762584.pdf |
| VISHAY | CRCW04024K70FKEDC | R7,R6,R28 | 3 | 4K7 0402 resistor | 0402 | SMD | T | | https://www.mouser.es/datasheet/2/427/crcwce3-1762584.pdf |
| VISHAY | CRCW04021K00FKEDC | R8 | 1 | 1K 0402 resistor | 0402 | SMD | T | | https://www.mouser.es/datasheet/2/427/crcwce3-1762584.pdf |
| VISHAY | CRCW040222R0FKEDC | R13,R14,R15,R16,R17,R18 | 6 | 22R 0402 resistor | 0402 | SMD | T | | https://www.mouser.es/datasheet/2/427/crcwce3-1762584.pdf |
| VISHAY | CRCW0402180RFKEDC | R19 | 1 | 180R 0402 resistor | 0402 | SMD | T | | https://www.mouser.es/datasheet/2/427/crcwce3-1762584.pdf |
| E-SWITCH | LL3301DF065QJ | SW2 | 1 | Switch 4 pin SPST | 5,9mm*3,9mm | SMD | T | | https://www.mouser.es/datasheet/2/140/P010454B-267753.pdf |
| SILICON LABS | CP2104-F03-GM | U1 | 1 | I/O Controller Interface IC USB to UART Bridge C | QFN24 | SMD | T | Y | https://www.mouser.es/datasheet/2/368/cp2104-1397921.pdf |
| MICROCHIP | MCP73832T-2DFI/OT | U2 | 1 | ry Management 4.2V Li-Ion/Li-Poly Chrg mgnt con | SOT-23-5 | SMD | T | | https://www.mouser.es/datasheet/2/268/20001984g-846362.pdf |
| DIODES | AP2112K-3.3 | U3 | 1 | oltage Regulators 600mA CMOS LDO 50mA 3.3V | SOT-23-5 | SMD | T | | https://www.mouser.es/datasheet/2/115/AP2112-271550.pdf |
| BOSCH | BME680 | U4 | 1 | Environmental Sensor VOC IIR EMC | LGA8 | SMD | T | Y | https://www.mouser.es/datasheet/2/783/BST-BME680-DS001-1509608.pdf |
| AMPHENOL | 93992-436HLF | J8 | 1 | pin socket 13 pin 2,54 mm male | 2,54mm | TH | | | https://www.amphenol-icc.com/media/wysiwyg/files/documentation/datasheet/boardwiretoboard/bwb_bergstik.pdf |
| 3M | 929974-01-26-RK | J3 | 1 | pin socket 3 pin 2,54 mm female | 2,54mm | TH | | | https://www.mouser.es/datasheet/2/1/ts0815-31700.pdf |
| AMPHENOL | 93992-436HLF | J9 | 1 | pin socket 3 pin 2,54 mm male | 2,54mm | TH | | | https://www.amphenol-icc.com/media/wysiwyg/files/documentation/datasheet/boardwiretoboard/bwb_bergstik.pdf |
| AVX | SD0805S020S0R5 | D1 | 1 | Schottky Diodes & Rectifiers 20V 0.5A SIZE 0805 | 0805 | SMD | T | Y | https://www.mouser.es/datasheet/2/40/schottky-776407.pdf |
| Johanson | 0868AT43A0020E | ANT1 | 1 | Ceramic Antennas ISM 868MHz | 7mm*2mm | SMD | T | Y | https://www.mouser.es/datasheet/2/611/0868AT43A0020-1519301.pdf |
| C&K | PCM12SMTR | SW1 | 1 | Slide Switches 0.3A SPDT ON-ON | ND | SMD/SMT | T | | https://www.mouser.es/datasheet/2/60/pcm-1382855.pdf |



ANT1
J2
R16
R17
R13
R14
R15
R18
R29
R10
C8
C7
C9
U5
J8
J7
R12
R11
U4
J5
J6
SW3
R21
C11
R24
Q3
R26
R23
R22
Q2
C12
R25
R27
D8
R20
C10
U6
J4
R28
R6
SW2
R9
U3
C5
C6
C13
R7
C2
U2
U1
3V3
D3
D4
D2
D1
C1
R5
D6
R8
C4
VBAT
D5
C3
J1
VBUS
SW1
J3
J9
R19
D7
Q1
BZ1

R1
R2
R4
R3
UOC
TFM 2020 E.P.G
BT1

# ANEXO B: CODIGO DE PROGRAMACION DE DISPOSITIVOS

# B1. CODIGO NODO/SENSOR

Código en Arduino/C para la configuración del nodo/sensor del sistema de emergencias. Tomará medidas con el sensor BME680 cada 10 segundos, si existe algún valor por encima del valor configurado como umbral. Se enviará la notificación al cliente BLE al que este conectado.

```
/***************************************************************************
  This is a library for the BME680 gas, humidity, temperature & pressure sensor

  Designed specifically to work with the Adafruit BME680 Breakout
  ----> http://www.adafruit.com/products/3660

  These sensors use I2C or SPI to communicate, 2 or 4 pins are required
  to interface.

  Adafruit invests time and resources providing this open source code,
  please support Adafruit and open-source hardware by purchasing products
  from Adafruit!

  Written by Limor Fried & Kevin Townsend for Adafruit Industries.
  BSD license, all text above must be included in any redistribution
 ***************************************************************************/
// IAQ part from code property David Bird 2019 https://github.com/G6EJD/BME680-Example/blob/master/ESP32_bme680_CC_demo_03.ino

#include <Wire.h>
#include <SPI.h>
#include <Adafruit_Sensor.h>
#include "Adafruit_BME680.h"
#include <BLEDevice.h>
#include <BLEServer.h>
#include <BLEUtils.h>
#include <BLE2902.h>
#include <thread>
#include <chrono>

#ifdef PCB_COMERCIAL
#define buttonPin 37
#else
//#define buttonPin 36//BUG ESP32 when it is used ADC1 I36 is pulled down for 80 ns
#define buttonPin 34
#endif
//node defines
#define NODE_ID 0x01
#define NODE_NAME "NODE1"
// pins
#define vbatPin 35
```

```
#define BME_SCK 13
#define BME_MISO 12
#define BME_MOSI 11
#define BME_CS 10
// BLE defines
//#define SERVICE_UUID       "4fafc201-1fb5-459e-8fcc-c5c9c331914b" //NETWORK1
#define SERVICE_UUID "d5cebf37-0739-41e1-94a4-b09555951919"//NETWORK2
#define CHARACTERISTIC_UUID "beb5483e-36e1-4688-b7f5-ea07361b26a8"
//sensor defines
#define TEMPERATURE_LIMIT_ALARM 30// valor de test
#define IAQ_LIMIT_ALARM 420//valor de test
#define VBAT_LIMIT_ALARM 3.75
#define SEALEVELPRESSURE_HPA (1013.25)
#define NUMBER_OF_PRESS 5

#define FLAG_TEMPERATURE_ALARM 0x01
#define FLAG_GAS_ALARM 0x02
#define FLAG_BATTERY_ALARM 0x04
#define FLAG_BUTTON_ALARM 0x08
//Globals variables
BLEServer* pServer = NULL;
BLECharacteristic* pCharacteristic = NULL;
bool deviceConnected = false;
bool oldDeviceConnected = false;
uint8_t value[2] = {NODE_ID,0};
float VBAT =0;
Adafruit_BME680 bme; // I2C

float hum_weighting = 0.01; // so hum effect is 25% of the total air quality score
float gas_weighting = 0.99; // so gas effect is 75% of the total air quality score

int   humidity_score, gas_score;
float gas_reference = 2500;
float hum_reference = 40;
int   getgasreference_count = 0;
int   gas_lower_limit = 10000;  // Bad air quality limit
int   gas_upper_limit = 300000; // Good air quality limit
int  IAQ = 0;

// the number of the LED pin
const int ledPin = 25;  // 25 corresponds to GPIO25

// setting PWM properties
const int freq = 2000;
const int ledChannel = 0;
const int resolution = 8;

struct Button {
```

```
        const uint8_t PIN;
        uint32_t numberKeyPresses;
        bool pressed;
        bool buttonPushed;
};

Button buttonAlarm = {buttonPin, 0, false,false};
void IRAM_ATTR isr() {

        buttonAlarm.numberKeyPresses += 1;
        buttonAlarm.pressed = true;
}
//temporal constants
const auto hundred_sec = std::chrono::seconds
{
        100
};
const auto ten_sec = std::chrono::seconds
{
        10
};
const auto one_sec = std::chrono::seconds
{
        1
};

void GetGasReference() {

        int readings = 10;
        for (int i = 1; i <= readings; i++) { // read gas for 10 x 0.150mS = 1.5secs
                gas_reference += bme.readGas();
        }
        gas_reference = gas_reference / readings;

}

String CalculateIAQ(int score) {
        String IAQ_text = "air quality is ";
        int temp = (100 - score) * 5;
        IAQ = temp;
        if     (temp >= 301)              IAQ_text += "Hazardous";
        else if (temp >= 201 && temp <= 300 ) IAQ_text += "Very Unhealthy";
        else if (temp >= 176 && temp <= 200 ) IAQ_text += "Unhealthy";
        else if (temp >= 151 && temp <= 175 ) IAQ_text += "Unhealthy for Sensitive Groups";
        else if (temp >=  51 && temp <= 150 ) IAQ_text += "Moderate";
        else if (temp >=  00 && temp <=  50 ) IAQ_text += "Good";
        Serial.print("IAQ Score = " + String(temp) + ", ");
        return IAQ_text;
}
```

```
int GetHumidityScore() {  //Calculate humidity contribution to IAQ index
          float current_humidity = bme.readHumidity();
          if (current_humidity >= 38 && current_humidity <= 42) // Humidity +/-5% around optimum
                    humidity_score = 0.25 * 100;
          else
          { // Humidity is sub-optimal
                    if (current_humidity < 38)
                        humidity_score = 0.25 / hum_reference * current_humidity * 100;
                    else
                    {
                        humidity_score = ((-0.25 / (100 - hum_reference) * current_humi-
                    dity) + 0.416666) * 100;
                     }
          }
          return humidity_score;
}

int GetGasScore() {
          //Calculate gas contribution to IAQ index
          gas_score = (0.75 / (gas_upper_limit - gas_lower_limit) * gas_reference - (gas_lower_li-
          mit * (0.75 / (gas_upper_limit - gas_lower_limit)))) * 100.00;
           if (gas_score > 75) gas_score = 75; // Sometimes gas readings can go outside of expec-
          ted scale maximum
           if (gas_score <  0) gas_score = 0;  // Sometimes gas readings can go outside of expec-
          ted scale minimum
           return gas_score;
}
class MyServerCallbacks: public BLEServerCallbacks {
            void onConnect(BLEServer* pServer) {
              deviceConnected = true;
              BLEDevice::startAdvertising();
            };

            void onDisconnect(BLEServer* pServer) {
              deviceConnected = false;
            }
};

//function reads the battery level
void bat( void * parameter )
{
          while(true){
                    VBAT = (float)(analogRead(vbatPin)) / 4095*2*3.3*1.1;
                    Serial.println("Vbat = "); Serial.print(VBAT); Serial.println(" Volts");
                    std::this_thread::sleep_for(hundred_sec);
          }
}
```

```
//thread function that reads the number of presses on emergency button
void button( void * parameter )
{
	while(true){
		if (buttonAlarm.pressed) {
			buttonAlarm.pressed = false;
		}
		if (buttonAlarm.numberKeyPresses>NUMBER_OF_PRESS)
			buttonAlarm.buttonPushed = true;

		static uint32_t lastMillis = 0;
		if (millis() - lastMillis > 60000) {
			lastMillis = millis();
			buttonAlarm.numberKeyPresses=0;
		}
		std::this_thread::sleep_for(one_sec);
	}
}
// Reading sensor parameters each 10 seconds
void sensor( void * parameter )
{
	  // I2C bus init
	Serial.println(F("BME680 test"));
	Wire.begin(22,21);
	delay(1000);
	if (!bme.begin(0x77,&Wire)) {
		Serial.println("Could not find a valid BME680 sensor, check wiring!");
		while (1);
	}
	//sensor parameter configuration
	bme.setTemperatureOversampling(BME680_OS_8X);
	bme.setHumidityOversampling(BME680_OS_2X);
	bme.setPressureOversampling(BME680_OS_4X);
	bme.setIIRFilterSize(BME680_FILTER_SIZE_3);
	bme.setGasHeater(320, 150); // 320*C for 150 ms

	while(true){
	  if (! bme.performReading()) {
		Serial.println("Failed to perform reading :(");
		return;
	  }

	  Serial.print("Temperature = ");
	  Serial.print(bme.temperature);
	  Serial.println(" *C");

	  Serial.print("Pressure = ");
	  Serial.print(bme.pressure / 100.0);
```

```
            Serial.println(" hPa");

            Serial.print("Humidity = ");
            Serial.print(bme.humidity);
            Serial.println(" %");

            Serial.print("Gas = ");
            Serial.print(bme.gas_resistance / 1000.0);
            Serial.println(" KOhms");

            Serial.print("Approx. Altitude = ");
            Serial.print(bme.readAltitude(SEALEVELPRESSURE_HPA));
            Serial.println(" m");

            humidity_score = GetHumidityScore();
            gas_score      = GetGasScore();

            //Combine results for the final IAQ index value (0-100% where 100% is good quality air)
            float air_quality_score = humidity_score + gas_score;
            Serial.println(" comprised of " + String(humidity_score) + "% Humi
        dity and " + String(gas_score) + "% Gas");
            if ((getgasreference_count++) % 5 == 0) GetGasReference();
            Serial.println(CalculateIAQ(air_quality_score));
            Serial.println("-----------------------------------------------------------");
            std::this_thread::sleep_for(ten_sec);
          }
}

//Thread that controls the diferents alarms that cause an emergency state
void alarm( void * parameter )
{
          value[1]=0x00;

          while(true)
          {
              if(bme.temperature>TEMPERATURE_LIMIT_ALARM)
                    value[1]|=FLAG_TEMPERATURE_ALARM;
              if(IAQ>IAQ_LIMIT_ALARM)
                    value[1]|=FLAG_GAS_ALARM;
              if(VBAT<VBAT_LIMIT_ALARM)
                    value[1]|=FLAG_BATTERY_ALARM;
              if(buttonAlarm.buttonPushed==true){
                     value[1]|=FLAG_BUTTON_ALARM;
                    buttonAlarm.buttonPushed=false;
              }
              if (deviceConnected) {
                            if(value[1]!=0x00){

                        #ifndef PCB_COMERCIAL
```

```
                                ledcWrite(ledChannel, 128);
                        //PWM enabled
                        #endif
                        pCharacteristic->setValue((uint8_t*)&value, 2);
                        pCharacteristic->notify();
                        Serial.print("\n\rNode= ");
                        Serial.print(value [0]);
                        Serial.print("\n\rAlarm= ");
                        Serial.print(value [1]);
                        Serial.println("\n\rNotify");
                        value[1]=0x00;
                    }
                    else{
                        #ifndef PCB_COMERCIAL
                          ledcWrite(ledChannel, 0);
                        #endif
                    }
                    delay(1000);
                  }
                   std::this_thread::sleep_for(ten_sec);
  }
}
void setup() {
            // Serial port init
            //setCpuFrequencyMhz(20);
            Serial.begin(115200);
            while (!Serial);


            // PWM
            ledcSetup(ledChannel, freq, resolution);
            ledcAttachPin(ledPin, ledChannel);


            //BLUETOOTH

            BLEDevice::init(NODE_NAME);

            // Create the BLE Server
            pServer = BLEDevice::createServer();
            pServer->setCallbacks(new MyServerCallbacks());

            // Create the BLE Service
            BLEService *pService = pServer->createService(SERVICE_UUID);

            // Create a BLE Characteristic
            pCharacteristic = pService->createCharacteristic(
                        CHARACTERISTIC_UUID,
                        BLECharacteristic::PROPERTY_READ   |
```

```
                BLECharacteristic::PROPERTY_WRITE  |
                BLECharacteristic::PROPERTY_NOTIFY |
                BLECharacteristic::PROPERTY_INDICATE
    );

  // https://www.bluetooth.com/specifications/gatt/viewer?attributeXmlFile=org.blue-
tooth.descriptor.gatt.client_characteristic_configuration.xml
  // Create a BLE Descriptor
  pCharacteristic->addDescriptor(new BLE2902());

  // Start the service
  pService->start();

  // Start advertising
  BLEAdvertising *pAdvertising = BLEDevice::getAdvertising();
  pAdvertising->addServiceUUID(SERVICE_UUID);
  pAdvertising->setScanResponse(true);
  pAdvertising->setMinPreferred(0x06);  // set value to 0x00 to not advertise this parameter
  BLEDevice::startAdvertising();
  Serial.println("Waiting a client connection to notify...");

  xTaskCreate(
              bat,        /* Task function. */
              "bat",      /* String with name of task. */
              10000,          /* Stack size in bytes. */
              NULL,           /* Parameter passed as input of the task */
              1,            /* Priority of the task. */
              NULL);          /* Task handle. */
  xTaskCreate(
              alarm,        /* Task function. */
              "alarm",      /* String with name of task. */
              10000,          /* Stack size in bytes. */
              NULL,           /* Parameter passed as input of the task */
              1,            /* Priority of the task. */
              NULL);          /* Task handle. */
  xTaskCreate(
              sensor,       /* Task function. */
              "sensor",     /* String with name of task. */
              10000,          /* Stack size in bytes. */
              NULL,           /* Parameter passed as input of the task */
              1,            /* Priority of the task. */
              NULL);          /* Task handle. */
  xTaskCreate(
              button,       /* Task function. */
              "button",     /* String with name of task. */
              10000,          /* Stack size in bytes. */
              NULL,           /* Parameter passed as input of the task */
              1,            /* Priority of the task. */
              NULL);          /* Task handle. */
```

```
        pinMode(buttonAlarm.PIN, INPUT_PULLUP);
        attachInterrupt(buttonAlarm.PIN, isr, FALLING);
}

void loop() {
          // disconnecting
          if (!deviceConnected && oldDeviceConnected) {
                    delay(500); // give the bluetooth stack the chance to get things ready
                    pServer->startAdvertising(); // restart advertising
                    Serial.println("start advertising");
                    oldDeviceConnected = deviceConnected;
          }
          // connecting
          if (deviceConnected && !oldDeviceConnected) {
                    // do stuff here on connecting
                    oldDeviceConnected = deviceConnected;
          }

         // Serial.println();
         delay(2000);
}
```

## B2. CODIGO GATEWAY/LORA

Código en Arduino/C para la configuración del gateway LoRa existente en cada vivienda. El código configurará al dispositivo como cliente BLE para que realice cada cierto periodo de tiempo un escaneado de los dispositivos disponibles. Si existe algún dispositivo con el UUID de servicio de su red, se conectará a él. Si recibe alguna notificación procedente de alguno de los dispositivos conectados, inmediatamente después habilitara el transceptor LoRa y generara la trama LoRa RAW para su envío.

```
#include <SPI.h>
#include <LoRa.h>
#include <Wire.h>
#include "SSD1306.h"
#include "BLEDevice.h"
#include <thread>
#include <chrono>
#include "SSD1306.h"
//define const
#define vbatPin 35
#ifdef PCB_COMERCIAL
//prototype based on comercial kit
#define SCK     5    // GPIO5  -- SX1278's SCK
#define MISO    19   // GPIO19 -- SX1278's MISnO
#define MOSI    27   // GPIO27 -- SX1278's MOSI
#define SS      18   // GPIO18 -- SX1278's CS
#define RST     23   // GPIO14 -- SX1278's RESET
#define DI0     26   // GPIO26 -- SX1278's IRQ(Interrupt Request)

#else//new designed model
#define SCK     18    // GPIO5  -- SX1278's SCK
#define MISO    19   // GPIO19 -- SX1278's MISnO
#define MOSI    23   // GPIO27 -- SX1278's MOSI
#define SS      5   // GPIO18 -- SX1278's CS
#define RST     33   // GPIO14 -- SX1278's RESET
#define DI0     14   // GPIO26 -- SX1278's IRQ(Interrupt Request)

#endif
#define BAND  868E6

#define NUMBER_OF_SERVERS 3
#define ID_GATEWAY 2
// The remote service we wish to connect to.
//static BLEUUID serviceUUID("4fafc201-1fb5-459e-8fcc-c5c9c331914b");//NETWORK1
static BLEUUID serviceUUID("d5cebf37-0739-41e1-94a4-b09555951919");//NETWORK2

// The characteristic of the remote service we are interested in.
```

```
static BLEUUID    charUUID("beb5483e-36e1-4688-b7f5-ea07361b26a8");

const auto thounsand_sec = std::chrono::seconds
{
   10000
};
//global variables
unsigned int counter = 0;
float VBAT =0;
#ifdef PCB_COMERCIAL
            SSD1306 display(0x3c, 21, 22);
#else
            SSD1306 display(0x3C, 22, 21);
#endif

String rssi = "RSSI --";
String packSize = "--";
String packet ;
uint8_t LoRa_msg[11];

int indice=0;
static boolean doConnect[NUMBER_OF_SERVERS];
static boolean connected[NUMBER_OF_SERVERS];
static boolean doScan[NUMBER_OF_SERVERS];
static BLERemoteCharacteristic* pRemoteCharacteristic;
static BLEAdvertisedDevice* myDevice[NUMBER_OF_SERVERS];
String line1_display;
String line2_display;
String line3_display;
String line4_display;
int device_connected;

// function which generate checksum of data stream
uint8_t Comp2checksum(const byte *data, size_t dataLength)
{
            uint8_t value = 0;
            for (size_t i = 0; i < dataLength; i++)
            {
              value += (uint8_t)data[i];
            }
            return (~value);
}
//notify callback from server BLE
static void notifyCallback(
            BLERemoteCharacteristic* pBLERemoteCharacteristic,
            uint8_t* pData,
            size_t length,
            bool isNotify) {
```

```
            Serial.print("\n\rData notified: ");
            Serial.printf("\n\rNode: 0x%02X",pData[0]);
            Serial.printf("\n\rAlarm: 0x%02X",pData[1]);
            counter++;
            // send packet
            //ID_SERVICE
            LoRa.idle();
            LoRa_msg[0]=0x12;
            LoRa_msg[1]=0x45;
            LoRa_msg[2]=0x89;
            LoRa_msg[3]=0x66;
            //ID_NETWORK
            LoRa_msg[4]=0x00;
            LoRa_msg[5]=0x01;
            //ID_GATEWAY
            LoRa_msg[6]=ID_GATEWAY;
            //ID_NODE
            LoRa_msg[7]=pData[0];
            //ID_GATEWAY
            LoRa_msg[8]=counter;
            //ALARM
            LoRa_msg[9]=pData[1];
            //CHECKSUM
            int tmp=Comp2checksum(LoRa_msg,10);

            LoRa_msg[10]=tmp;


          line2_display="Senging packet: "+ String(counter);
          line3_display="Node: "+ String(pData[0]);
          line4_display="Alarm: "+ String(pData[1]);

            LoRa.beginPacket();
            for(int i=0;i<11;i++){

                  LoRa.print(String(LoRa_msg[i]));
                  LoRa.print("/");
            }

            LoRa.endPacket();
            delay(10);
            LoRa.sleep();

}

class MyClientCallback :

          public BLEClientCallbacks {
            void onConnect(BLEClient* pclient) {
```

```
        }

        void onDisconnect(BLEClient* pclient) {
          Serial.println("onDisconnect");
           xTaskCreate(
                    scan,        /* Task function. */
                    "scan",       /* String with name of task. */
                    10000,          /* Stack size in bytes. */
                    NULL,          /* Parameter passed as input of the task */
                    1,            /* Priority of the task. */
                    NULL);          /* Task handle. */
        }
};

bool connectToServer(int id_device) {
        Serial.print("Forming a connection to ");
        Serial.println(myDevice[id_device]->getAddress().toString().c_str());

        BLEClient*  pClient  = BLEDevice::createClient();
        Serial.println(" - Created client");
        device_connected++;
        pClient->setClientCallbacks(new MyClientCallback());
        line1_display="Node Connected: " + String(device_connected);
        // Connect to the remove BLE Server.
        pClient->connect(myDevice[id_device]);
        Serial.println(" - Connected to server");

        // Obtain a reference to the service we are after in the remote BLE server.
        BLERemoteService* pRemoteService = pClient->getService(serviceUUID);
        if (pRemoteService == nullptr) {
                Serial.print("Failed to find service UUID: ");
                Serial.println(serviceUUID.toString().c_str());
                pClient->disconnect();
                return false;
        }
        Serial.println(" - Founded  service");


        // Obtain a reference to the characteristic in the service of the remote BLE server.
        pRemoteCharacteristic = pRemoteService->getCharacteristic(charUUID);
        if (pRemoteCharacteristic == nullptr) {
          Serial.print("Failed to find  characteristic UUID: ");
          Serial.println(charUUID.toString().c_str());
          pClient->disconnect();
          return false;
        }
        Serial.println(" - Founded characteristic");

        // Read the value of the characteristic.
```

```
            if(pRemoteCharacteristic->canRead()) {
                        std::string value = pRemoteCharacteristic->readValue();
                        const char *val = value.c_str();
                        // Serial.print("The characteristic value was: ");
                        // Serial.println(value.c_str());
                        //  for (int i = 0; i < 4; i++) {
                        //     Serial.print((int)val[i], HEX);
                        //     Serial.print(" ");
                        //  }
            }

            if(pRemoteCharacteristic->canNotify())
              pRemoteCharacteristic->registerForNotify(notifyCallback);

            connected[id_device] = true;
}
/**
 * Scan for BLE servers and find the first one that advertises the service we are looking for.
 */
class MyAdvertisedDeviceCallbacks: public BLEAdvertisedDeviceCallbacks {
 /**
   * Called for each advertising BLE server.
   */
  void onResult(BLEAdvertisedDevice advertisedDevice) {
    Serial.print("\n\rBLE Advertised Device found: ");
    Serial.println(advertisedDevice.toString().c_str());

    // We have found a device, let us now see if it contains the service we are looking for.
    if (advertisedDevice.haveServiceUUID() && advertisedDevice.isAdvertisingService(serviceUUID)) {

     //
                myDevice[indice] = new BLEAdvertisedDevice(advertisedDevice);
                doConnect[indice] = true;
                indice++;

                doScan[indice] = true;
                if(indice>=NUMBER_OF_SERVERS)
                      BLEDevice::getScan()->stop();

            } // Found our server
} // onResult
}; // MyAdvertisedDeviceCallbacks


void setup() {

            setCpuFrequencyMhz(80);
            //START LORA
            pinMode(16,OUTPUT);
```

```
 pinMode(23,OUTPUT);
 device_connected=0;
// #ifdef PCB_COMERCIAL
 digitalWrite(16, LOW);    // set GPIO16 low to reset OLED
 delay(50);
 digitalWrite(16, HIGH); // while OLED is running, must set GPIO16 in high
 //#endif
 Serial.begin(115200);
 while (!Serial);
 Serial.println();
 Serial.println("LoRa Gateway");

 SPI.begin(SCK,MISO,MOSI,SS);
 LoRa.setPins(SS,RST,DI0);
 if (!LoRa.begin(868E6)) {
            Serial.println("Starting LoRa failed!");
            while (1);
 }
 LoRa.sleep();
 //END LORA
 //INIT BLE CLIENT
 for(int x=0;x<NUMBER_OF_SERVERS;x++){
             doConnect[x]= false;
             connected[x]= false;
             doScan[x]= false;
 }
  BLEDevice::init("");

 // Retrieve a Scanner and set the callback we want to use to be informed when we
 // have detected a new device.  Specify that we want active scanning and start the
 // scan to run for 5 seconds.
 BLEScan* pBLEScan = BLEDevice::getScan();
 pBLEScan->setAdvertisedDeviceCallbacks(new MyAdvertisedDeviceCallbacks());
 pBLEScan->setInterval(1349);
 pBLEScan->setWindow(449);
 pBLEScan->setActiveScan(true);
 pBLEScan->start(5, true);
 //END BLE CLIENT

 Serial.println("init ok");
 display.init();
 display.flipScreenVertically();
 display.setFont(ArialMT_Plain_10);
 line1_display="Init";
 line2_display="UOC";
 line3_display="Gateway";
 line4_display="LoRa";

//Battery
```

```
        xTaskCreate(
                    bat,        /* Task function. */
                    "bat",      /* String with name of task. */
                    10000,          /* Stack size in bytes. */
                    NULL,           /* Parameter passed as input of the task */
                    1,            /* Priority of the task. */
                    NULL);          /* Task handle. */
        delay(1500);
}
//refresh data showed on OLED
void writeCommand(String line1,String line2, String line3,String line4)
{
        display.clear();
        display.setTextAlignment(TEXT_ALIGN_LEFT);
        display.setFont(ArialMT_Plain_10);
        display.drawString(0, 0, line1);
        display.drawString(0, 20, line2);
        display.drawString(0, 30, line3);
        display.drawString(0, 40, line4);
        display.display();
        delay(1500);
}

void scan( void * parameter )
{
        Serial.println("Scanning");
        BLEScan* pBLEScan = BLEDevice::getScan();
        pBLEScan->setAdvertisedDeviceCallbacks(new MyAdvertisedDeviceCallbacks());
        pBLEScan->setInterval(1349);
        pBLEScan->setWindow(449);
        pBLEScan->setActiveScan(true);
        pBLEScan->start(5, true);
        for(int x=0;x<NUMBER_OF_SERVERS;x++){
                    doConnect[x]= false;
                    connected[x]= false;
                    doScan[x]= true;
        }
        vTaskDelete( NULL );
}
void bat( void * parameter )
{
        while(true){
                    VBAT = (float)(analogRead(vbatPin)) / 4095*2*3.3*1.1;
                    Serial.printf("\n\rVbat = %f Volts",VBAT);
                    std::this_thread::sleep_for(thounsand_sec);
        }
}
void loop() {
        //#ifdef PCB_COMERCIAL
```

```
        writeCommand(line1_display,line2_display,line3_display,line4_display);
      //#endif
        for(int i=0;i<indice;i++)
        {
                if (doConnect[i] == true) {
                        if (connectToServer(i)) {
                                Serial.println("\n\rConnected to the BLE Server.");
                        } else {
                                Serial.println("\n\rFail: Not connected to the BLE Server");
                        }
                        doConnect[i] = false;
                }
        }
        delay(1000);   // wait for a second
}
```

# B3. CODIGO GATEWAY SIGFOX

Código en micropython para la configuración del dispositivo gateway Sigfox. Encargado de enviar la información del sistema de emergencias a la nube.
Esta formado principalmente por una maquina de estados que controlara dos sockets, uno para LoRa y otro para Sigfox. Su estado natural después del a inicialización será la de escuchar toda la información recibida LoRa, si alguna de las tramas recibidas dispone de la cabecera de identificación del servicio de emergencias, esta se procesará y se verificará su integridad generando un checksum que debe coincidir con el ultimo byte de la trama.
Si se confirma que la trama es correcta, se deshabilitará el socket LoRa para habilitar el Socket Sigfox y enviar la información a la nube.

```
from network import Sigfox
import struct
import pycom
from network import LoRa
import socket
import _thread
import time
import array
import re
#Machine States
INIT = 0
LISTEN = 1
TX_SIGFOX = 2
RECOVER =3

# # make the socket blocking
stateMachine =INIT
while(True):
        #State Start and init LoRa communication, listening mode and power mode always on
          if stateMachine ==INIT
                    lora = LoRa(mode=LoRa.LORA, region=LoRa.EU868 ,frequency= 868000000, band-
              width=LoRa.BW_125KHZ, sf=7, preamble=8,
                    coding_rate=LoRa.CODING_4_5, power_mode=LoRa.ALWAYS_ON,
                    tx_iq=False, rx_iq=False, public=False,device_class=LoRa.CLASS_C)
                    s_lora = socket.socket(socket.AF_LORA, socket.SOCK_RAW)
                    stateMachine = LISTEN
        #State Listening, filter messages that not belong to the emergency service, alfter that check the
        message quality
          elif stateMachine == LISTEN:
                    time.sleep(1)
                    print('Running listening status')
                    response = s_lora.recv(64)
                    print(response)
                    header = response[:13]
```

```
                    list_data =[]
                    if header == b'18/69/137/102':
                                cadena = response.decode("utf-8", "ignore")
                                print(cadena)
                                list_data = cadena.split("/")
                                checksum_msg = list_data[0:10]
                                print(list_data)
                                print('ID service correct')
                                value = 10
                                value = 0
                                for x in checksum_msg:
                                            value+=int(x,10)
                                value=255-(value%256)
                                print(value)
                                tmp = int(list_data[10],10)
                                print(tmp)

                                if tmp==value:
                                    print('CHK correct')
                                    temp2 =list_data[4:10]
                                    for i in range(0, len(temp2)):
                                       temp2[i] = int(temp2[i])
                                    send_sigfox=bytes(temp2)
                                    stateMachine = TX_SIGFOX
        # Transmission   Sigfox message state, previously disable LoRa socket because both use same
        transceiver
          elif stateMachine == TX_SIGFOX:
                    lora.power_mode(LoRa.SLEEP)
                    s_lora.close()
                    lora.nvram_save()
                    sigfox = Sigfox(mode=Sigfox.SIGFOX, rcz=Sigfox.RCZ1)
                    print('sending data throught Sigfox')
                    # create a Sigfox socket
                    s = socket.socket(socket.AF_SIGFOX, socket.SOCK_RAW)
                    s.setsockopt(socket.SOL_SIGFOX, socket.SO_RX, False)
                    s.setblocking(True)
                    s.settimeout(60)
                    s.send(send_sigfox)
                    stateMachine = RECOVER
          else: #stateMachine == RECOVER:
                  stateMachine =INIT

s.close()
pycom.heartbeat(False)
```

# ANEXO C: PROGRAMACION SERVIDOR

# C1. INSTANCIAS NODE.JS

## INSTANCIA APP.JS

Código en node.js para la recepción de las alarmas procedentes del servidor Sigfox. La instancia estará continuamente en ejecución escuchando el puerto 8080. Cuando se reciba una alarma procederá a la extracción de la información relevante y realizara una consulta a la base de datos direcciones para ver si existe algún dispositivo ID y gateway ID coincidente. Si es así, la información de la dirección se añadirá a la información de la alarma y se almacenará en la colección de alarmas.

```
var express = require("express");
var app = express();
var bodyParser = require("body-parser");
var mongo = require('mongodb');
var MongoClient = mongo.MongoClient;
var url = "mongodb://localhost:27017/sigfoxdb";
var cors = require('cors');
app.use(cors());
app.use(bodyParser.urlencoded({ extended: false }));
app.use(bodyParser.json());
app.post('/push', function (req, res) {
	console.log(req.body);
	body = JSON.parse(Object.keys(req.body)[0])
	var deviceId = body.device;
	var date = new Date();
	var data = body.data;
	console.log(typeof(data) );
	var network = parseInt(data.slice(0,4));
	var gateway = parseInt(data.slice(4,6));
	var node = parseInt(data.slice(6,8));
	var count = parseInt(data.slice(8,10));
	var alarm = parseInt(data.slice(10,12));
	var seqNumber = parseInt(body.seqNumber);
	MongoClient.connect(url, function(err, db) {
		if (err) throw err;
		var dbo = db.db("sigfoxdb");
		var query = {deviceId:deviceId,gateway:gateway.toString()};
		var mysort = { time: 1 };
		dbo.collection("address").find(query).sort(mysort).toArray(function(err, result) {
			if (err) throw err;
			var alarmArray ={
				deviceId: deviceId,
				date: date,
				network: network,
				gateway: gateway,
				node: node,
				count: count,
				alarm: alarm,
				seqNumber: seqNumber,
				address : result[0].address,
			};
			console.log(result);
```

```
                                console.log(alarmArray);
                                dbo.collection("alarms").insertOne(alarmArray, function(err, result2) {
                                if (err) throw err;
                                console.log(" documents inserted");
                                db.close();
                                res.sendStatus(200);
                                });
                        });
        });
});
app.listen(8080, function() {
        console.log("Node server running on {SERVER INSTANCE PUBLIC IP}:8080");
});
```

---

---

---

## INSTANCIA APP  WEB.JS

Código en node.js para la atención de las peticiones procedentes de la aplicación web del cliente. Cuando se instancie realizara una escucha continua al puerto 3000

```
var express = require("express"),
app = express(),
bodyParser = require("body-parser"),
methodOverride = require("method-override");
var mongo = require('mongodb');
var MongoClient = mongo.MongoClient;
var url = "mongodb://localhost:27017/sigfoxdb";
var fs = require('fs');
var engine = require('consolidate');
app.set('views', __dirname + '/public/views');
app.engine('html', require('ejs').renderFile);
app.set('view engine', 'html');
app.use(bodyParser.urlencoded({ extended: false }));
app.use(bodyParser.json());
app.use(methodOverride());
app.use(express.static('views'));
var router = express.Router();

router.get('/list', function(req, res) {
      MongoClient.connect(url, function(err, db) {
          if (err) throw err;
          var dbo = db.db("sigfoxdb");
          var allAlarms = [];
          var query = {};
          var mysort = { time: 1 };
          dbo.collection("alarms").find(query).sort(mysort).toArray(function(err, result) {
                  if (err) throw err;
                  db.close();
                  result.forEach(function(alarm){
                  var elem = new Object();
                  elem["deviceId"]=alarm.deviceId;
                  elem["date"]=alarm.date;
```

```
                elem["network"]=alarm.network;
                elem["gateway"]=alarm.gateway;
                elem["node"]=alarm.node;
                elem["count"]=alarm.count;
                elem["alarm"]=alarm.alarm;
                elem["seqNumber"]=alarm.seqNumber;
                elem["address"]=alarm.address;
                allAlarms.push(elem);
                console.log(elem)
        });
        res.render(__dirname + '/html/alarm.html',{alarms:result});
    });
    })
});

router.get('/list_address', function(req, res) {
        MongoClient.connect(url, function(err, db) {
                if (err) throw err;
                var dbo = db.db("sigfoxdb");
                var allAddress = [];
                var query = {};
                var mysort = { time: 1 };
                dbo.collection("address").find(query).sort(mysort).toArray(function(err, result) {
                        if (err) throw err;
                        db.close();
                        result.forEach(function(address){
                                var elem = new Object();
                                elem["deviceId"]=address.deviceId;
                                elem["gateway"]=address.gateway;
                                elem["address"]=address.address;
                                allAddress.push(elem);
                                console.log(elem)
                        });
                res.render(__dirname + '/html/address.html',{address:result});
                });
        })
});


router.get('/', function(req, res) {
        res.sendFile('html/main.html', {root: __dirname })
});


router.delete('/delete_alarms', function(req, res) {
        MongoClient.connect(url, function(err, db) {
                var query = {};
                var dbo = db.db("sigfoxdb");
                dbo.collection("alarms").remove(query,function(err, result) {
                        if (err) throw err;
                        db.close();
                        res.send("Borrado registro de direcciones");
                });
        });
});
```

```
router.delete('/delete_address', function(req, res) {
	MongoClient.connect(url, function(err, db) {
		var query = {};
		var dbo = db.db("sigfoxdb");
		dbo.collection("address").remove(query,function(err, result) {
			if (err) throw err;
			db.close();
			res.send("Borrado registro de direcciones");
		});
	});
});


router.post('/registro', function(req, res) {
	var body =req.body;
	var insert = false;


	var address ={
		deviceId: body.device_id,
		gateway: body.gateway_id,
		address: body.address_device,
	};
	console.log(address);
	MongoClient.connect(url, function(err, db) {
		if (err) throw err;
		var dbo = db.db("sigfoxdb");
		var query = { deviceId:address.deviceId,gateway:address.gateway};
		dbo.collection("address").find(query).toArray(function(err, result) {
		if (err) throw err;
		db.close();
		console.log(result.length);
		if (result.length <= 0) {
			MongoClient.connect(url, function(err, db) {
			if (err) throw err;
			var dbo = db.db("sigfoxdb");
			dbo.collection("address").insertOne(address, function(err, result) {
			if (err) throw err;
			console.log(" documents inserted");
			db.close();
			res.send("Direccion no encontrada. Añadida nueva");
			});
			});
		}
		else{
			MongoClient.connect(url, function(err, db) {
			if (err) throw err;
			var dbo = db.db("sigfoxdb");
			var query = { deviceId:address.deviceId,gateway:address.gateway};
			dbo.collection("address").update(query,address,function(err, result) {
			if (err) throw err;
			db.close();
			res.send("Direccion encontrada. Modificada existente");
			});
			});
		}
		});
```

```
	});
});


router.delete('/baja_registro', function(req, res) {
	var body =req.body;
	var insert = false;
	var address ={
		deviceId: body.device_id,
		gateway: body.gateway_id,
		address: body.address_device,
	};
	console.log(address);
	MongoClient.connect(url, function(err, db) {
		if (err) throw err;
		var dbo = db.db("sigfoxdb");
		var query = { deviceId:address.deviceId,gateway:address.gateway};
		dbo.collection("address").find(query).toArray(function(err, result) {
		if (err) throw err;
		db.close();
		console.log(result.length);
		if (result.length <= 0) {
			res.send("Direccion no encontrada!");
		}
		else{
			MongoClient.connect(url, function(err, db) {
			if (err) throw err;
			var dbo = db.db("sigfoxdb");
			var query = { deviceId:address.deviceId,gateway:address.gateway};
				dbo.collection("address").remove(query,address,function(err, result)
				{
					if (err) throw err;
					db.close();
					res.send("Borrado dispositivo asociado a direccion");
				});
			});
		}
	});
	});
});

router.get('/alta', function(req, res, next) {
	res.sendFile('html/alta.html', {root: __dirname })
// or res.send(html)
})
app.use(router);
app.listen(3000, function() {
	console.log("Node server running on http://localhost:3000");
});
```

# C2 visualización HTML/AJAX/JQUERY

## MAIN.HTML

Se trata del código necesario para el desarrollo del panel principal de la aplicación de visualización de alarmas. Esta consta de dos partes bien diferenciadas.
Script: Todas las acciones a realizar cuando se pulsa alguno de los botones de formulario. Se realizarán peticiones al servidor y el resultado se devolverá en la capa situada en la parte inferior del panel.
HTML: Se trata del código HTML con los formularios del panel de control de la aplicación, se pueden encontrar tres bloques. Consulta, Alta/modificación/Baja de un dispositivo o borrado de las bases de datos.

```
<!DOCTYPE html>
<html>
	<style>
		<!-- Configuracion de estilos -->
		body {
			background-color: rgb(255,255,255);
		}
		div{
			margin: center;
			width: 100%;
			border: 2px solid black;
			padding: 10px;
			background-color: rgb(220,220,220);
			border-color:rgb(0,0,0);
		}
		input {
			width: 200px;
			text-align: right;
		}
		label{
			display: inline-block;
			float: left;
			clear: left;
			width: 150px;
			text-align: LEFT;
		}
		button {
			background-color: #4CAF50; /* Green */
			border: none;
			color: white;
			padding: 15px 32px;
			text-align: center;
			text-decoration: none;
			display: inline-block;
			font-size: 16px;
			border: 2px solid #000000;
			width: 250px;
			transition-duration: 0.4s;
		}
		button:hover {
			background-color: #A0FFA0; /* Green */
```

```
                color: white;
        }
        table {
                border: 1px solid black;
        }
        th, td {
                padding: 15px;
                text-align: center;
                background-color:#C0C0C0;
        }
    </style>
<head>
    <script type='text/javascript'
    src="//ajax.googleapis.com/ajax/libs/jquery/1.8.3/jquery.min.js"></script>
    <!-- Acción sobre el botón con id=boton y actualizamos el div con id=capa -->
    <!-- Eventos para envio a servidor de todas las funciones get,post y delete -->
    <!-- Los elementos recibidos desde servidor se muestran en la capa contenido -->
    <script type="text/javascript">
    var flag=false;<!-- La variable flag deshabilita el refresco de alarmas -->
    $(document).ready(function() {
        $("#alta_registro").click(function(event){
            flag=false;
            $.ajax({
                type: "POST",
                url: "http://35.178.151.208:3000/registro",
                dataType: "html",
                data: $("#alta_form").serialize(),
                success: function( response) {
                    $('#contenido').html(response);
                    $(".response").css("color", "green");
                    $(".response").css("font-size", "40px");
            }
            });
        });
        $("#baja_registro").click(function(event){
            flag=false;
            $.ajax({
                type: "DELETE",
                url: "http://35.178.151.208:3000/baja_registro",
                dataType: "html",
                data: $("#alta_form").serialize(),
                success: function( response) {
                    $('#contenido').html(response);
                    $(".response").css("color", "green");
                    $(".response").css("font-size", "40px");
                }
            });
        });
        $("#baja_alarma").click(function(event) {
            flag=false;
            $.ajax({
                type: "DELETE",
                url: "http://35.178.151.208:3000/delete_alarms",
                success: function( response) {
                    $('#contenido').html(response);
                    $(".response").css("color", "green");
                    $(".response").css("font-size", "40px");
```

```
                        }
                    });
                });
                $("#baja_address").click(function(event) {
                    flag=false;
                    $.ajax({
                        type: "DELETE",
                        url: "http://35.178.151.208:3000/delete_address",
                        success: function( response) {
                            $('#contenido').html(response);
                            $(".response").css("color", "green");
                            $(".response").css("font-size", "40px");
                        }
                    });
                });
                $("#consulta_address").click(function(event) {
                    $("#contenido").load('http://35.178.151.208:3000/list_address');
                    $(".response").css("color", "black");
                    $(".response").css("font-size", "16px");
                    flag=false;
                });
                $("#consulta_alarma").click(function(event) {
                    flag=true;
                    $(".response").css("color", "black");
                    $(".response").css("font-size", "16px");
                    setInterval(function(){
                    if(flag ==true){
                        $("#contenido").load('http://35.178.151.208:3000/list');
                    }
                    }, 2000);<!-- actualizacion de funcion cada 2 segundos -->
                });
        });
</script>
</head>
<body>
        <div id="menu" style="width: 100%; height:60%;">
            <table style="width:100%;">
            <tr>
                <h1>SISTEMA DETECCION DE EMERGENCIAS<br></h1>
            </tr>
            <tr>
            <td style="width:30%;">
            <p><h2>CONSULTAS</h2></p>
            <form method="get" action="/list" ><br>
                <button type="button" id="consulta_alarma"> Alarmas</button>
            </form>
            <form method="get" action="/list_address" ">
                <button type="button" id="consulta_address"> Direcciones</button>
            </form>
            </td>
            <td style="width:30%;">
                <form method="post" action="/registro" id="alta_form">
                <p><h2>ALTA/MODIFICIACION/BAJA DISPOSITIVO</h2></p>
                <label for="device_id">ID Device: </label>
                <input type="text" id="device_id" name="device_id"><br>
                <label for="gateway_id" width =100px>ID Gateway: </label>
                <input type="text" id="gateway_id" name="gateway_id"><br>
```

```
                    <label for="address_device">Address Device:</label>
                    <input type="text" id="address_device" name="address_device"><br><br>
                    </form>
                    <button type="button" id="alta_registro">Alta/Modificación</button>
                    <button type="button" id="baja_registro">Baja</button><br>
                </td>
                <td style="width:30%;">
                    <p><h2>BORRAR</h2></p>
                    <form method="get" action="/delete_alarms" ><br>
                    <button type="button" id="baja_alarma"> Alarmas</button>
                    </form>
                    <form method="get" action="/delete_address ">
                    <button type="button" id="baja_address"> Direcciones</button>
                    </form>
                </td>
                </tr>
                </table>
        </div>
        <div id="contenido" class="response" "style="width: 100%; height:60%;"></div>
</body>
</html>
```

---

---

---

## ADDRESS.HTML

Código HTML para listar las direcciones existentes en la base de datos del servidor.

```
<html>
        <head>
        <title>List of Address</title>
        <h1>Address</h1>
        <!--<meta http-equiv="refresh" content="5">-->
        </head>
<body>
        <div>
                <form>
                <ul>
                <% address.forEach(function(address) { %>
                <li><b>Device ID: </b><%= address.deviceId %>
                    
                <b>Gateway ID:</b> <%= address.gateway %>     
                <b> Address: </b> <%= address.address %> </li>
                <% }); %>
                </ul>
                </form>
        </div>
</body>
</html>
```

## ALARMS.HTML

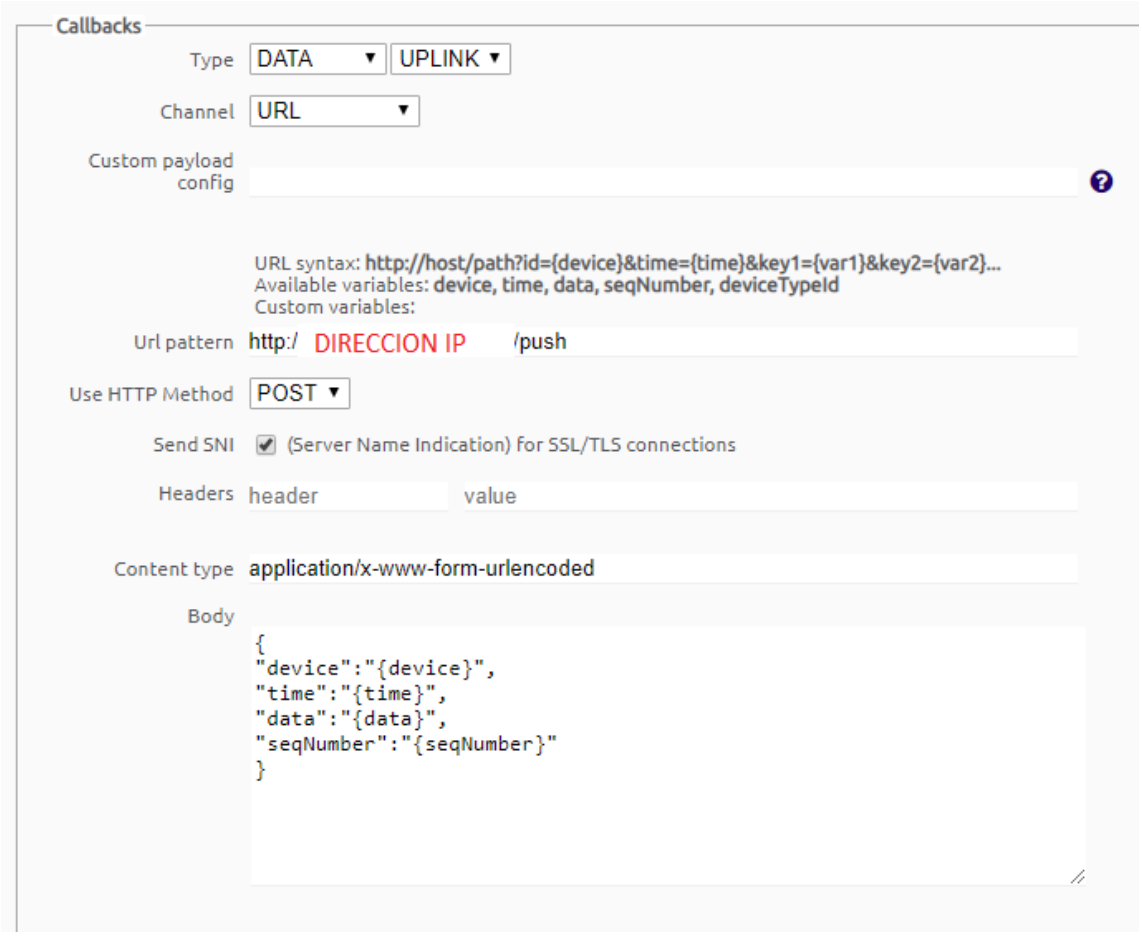
Código HTML para listar las alarmas existentes en la base de datos del servidor.

```
<html>
        <style>
        </style>
        <head>
        <title>List of Alarms</title>
        <h1>Alarms</h1>
        <!--<meta http-equiv="refresh" content="5"> -->
        </head>
        <body>
        <div>
                <form>
                        <ol>
                        <% alarms.forEach(function(alarm) { %>
                        <li><b>Device ID:</b> <%= alarm.deviceId %>
                             <b>Gateway ID:</b>
                        <%= alarm.gateway%>     
                        <b>Alarm Address:</b> <%= alarm.address %>     
                        <b>Date:</b> <%= alarm.date %>     
                        <b style =" color: red;"> Alarm Code: <%= alarm.alarm %> </b> </li>
                        <% }); %>
                        </ol>
                </form>
        </div>
        </body>
</html>
```

# ANEXO D: TUTORIAL DE INSTALACIÓN DE CALLBACK Y SERVIDOR

# D1. CONFIGURACIÓN DE CALLBACK EN SIGFOX

1. Lo primero que hay que hacer es registrar el dispositivo. Estas gestiones se realizan en https://backend.sigfox.com/ en el apartado de devices y acceder al apartado de New. Es necesario descargarse el firmware de actualización del dispositivo Pycom y conseguir los parámetros del dispositivo asociado, disponibles a través de dicho instalable. Estos parámetros identificativos y únicos de cada dispositivo son: Device ID, LoRa MAC, Sigfox ID y Sigfox PAC.
2. 
3. Tras haber dado de alta al dispositivo, podremos acceder a sus opciones de configuración. La opción que nos permite configurar como y a que dirección enviar esta información es la opción callbacks.
    a. Presionar pestaña *new*
    b. Seleccionar *Custom Callback*
    c. Realizar la siguiente configuración




NOTA: La dirección IP es la de destino, en este caso el Servidor AWS

# D2. PASOS PARA INSTALAR SERVIDOR

## 1. Configuración de instancia lightsail de AWS.

El servidor se ha desarrollado bajo una instancia lightsail Linux del servicio AWS. Es necesario realizar una serie de configuraciones previas a la ejecución de las apps.
Lo primero de todo es abrir los puertos que se van a utilizar

| | | | |
|---|---|---|---|
| SSH | TCP | 22 | Cualquier dirección IP<br>SSH/RDP del navegador Lightsail |
| HTTP | TCP | 80 | Cualquier dirección IP |
| HTTPS | TCP | 443 | Cualquier dirección IP |
| Personalizado | TCP | 3000 | Cualquier dirección IP |
| Personalizado | TCP | 8080 | Cualquier dirección IP |

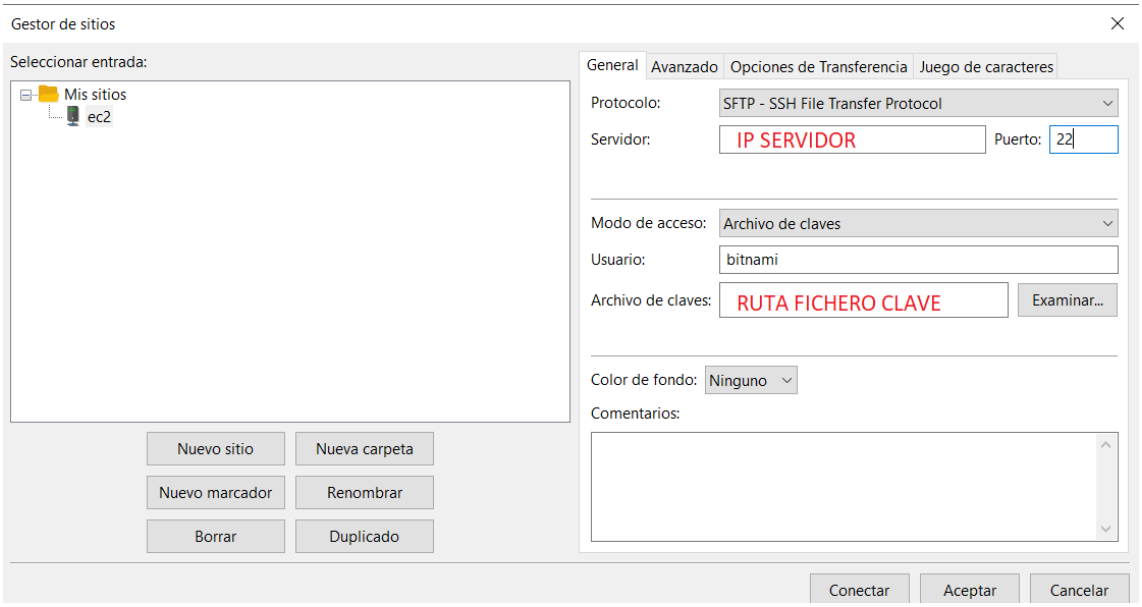
Algunos siempre vienen configurados por defecto, otros es necesario abrirlos manualmente.

- **SSH 22:** Se utilizará para abrir los ficheros a través de un cliente ftp como FileZilla, nos comunicaremos a través de SSH y podremos acceder al directorio.
- **TCP 8080:** Puerto para la app.js encargada de la recepción de mensajes procedentes del servidor de Sigfox.
- **TCP 3000:** Puerto para app_web.js se encargará de la gestión de peticiones procedente del cliente.

## 2. Configuración de cliente FTP FileZilla.

Este tipo de aplicaciones son imprescindibles para poder subir todos los ficheros importantes del proyecto. En este caso se recomienda FileZilla ya que permite la conexión con el servidor a través SFTP-SSH

En la siguiente imagen se puede ver la configuración del sitio realizada para este trabajo:




La clave de acceso utilizada es un fichero descargable desde la cuenta de usuario de AWS.

## 2. Configuración de servidor virtual.

1. Instalaciones necesarias para ejecutar apps
   **>npm install npm**
   **>npm install express**
   **>npm install mongodb**
   **>npm install ejs**
   **>npm install html**
   **>npm install method-override**
   **>npm install cors –save**

2. Creación de BBDD y colecciones:

   a. Ejecutar mongo
   **>mongo**

   b. Usar bbdd
   >**use sigfoxdb**

   c. Crear colección:
   **>db.createCollection(“alarms”)**
   **>db.createCollection(“address”)**

   d.Para borrar datos de una colección:
   **>use sigfox**
   **>db.”nombre”.drop()**

Si existe algún problema con mongo, intentar:

**>sudo service mongod stop**
**>sudo rm /var/lib/mongodb/mongod.lock**
**>mongod –repair**
**>sudo service mongod start**

3. Para mantener las instancias ejecutándose permanentemente es necesario utilizar forever o pm2, en este caso se recomienda el uso de la segunda:

a. instalación:
**>sudo npm install pm2@latest -g**

b. Asegurarse que está instalada:
**>which pm2** (debe responder la ruta donde está instalado)

c. Ejecutar app bajo pm2:
**>pm2 start app.js**

d. podemos ver las instancias ejecutándose
**>pm2 ls**

También se puede lanzar bajo node:

**> sudo node app.js**
**> sudo node app_web.js**