

TRABAJO FIN DE GRADO



UCAM
UNIVERSIDAD CATÓLICA
SAN ANTONIO

ESCUELA UNIVERSITARIA POLITÉCNICA

Grado en Ingeniería en Sistemas de Telecomunicación

Sistema IoT de medición de CO₂ para prevención de
COVID-19 basado en LoRaWAN

Autor:

Pablo del Arco Ortiz

Director:

Dr. Rafael Berenguer Vidal

Murcia, 15 de julio de 2021

TRABAJO FIN DE GRADO



ESCUELA UNIVERSITARIA POLITÉCNICA

Grado en Ingeniería en Sistemas de Telecomunicación

Sistema IoT de medición de CO₂ para prevención de
COVID-19 basado en LoRaWAN

Autor:

Pablo del Arco Ortiz

Director:

Dr. Rafael Berenguer Vidal

Murcia, 15 de julio de 2021

AGRADECIMIENTOS

Gracias a mi familia por apoyarme y animarme en cada momento, sin ellos no habría sido posible.

Gracias a mi tutor del TFG, Rafael Berenguer por haberme ayudado en este último proyecto.

Gracias a la UCAM y a sus magníficos profesores por haberme enseñado tanto en todo este tiempo. Gracias a todo lo aprendido en la carrera, he podido realizar este proyecto.

Gracias a Javier Fernández por ayudarme con la impresión 3D, sin su ayuda habría sido muy complicado crear una carcasa a medida.

Este TFG representa mucho para mí pues he adquirido grandes conocimientos que me permitirán afrontar cualquier reto en mi carrera profesional.

Gracias a todos.

RESUMEN

Las tecnologías relacionadas con el Internet de las Cosas están sentando las bases de una sociedad cada vez más digitalizada. En concreto, las redes LPWAN (Low Power Wide Area Network) están abriendo un nuevo horizonte de posibilidades donde prima el bajo consumo y un gran alcance.

Debido a la reciente pandemia de la COVID-19, cuyas consecuencias seguimos sufriendo en la actualidad, este proyecto busca reducir el riesgo de contagio mediante la medición de la concentración del CO₂ en espacios interiores. Para ello, el presente trabajo hace uso de la tecnología LoRaWAN para la interconexión de los nodos y la transmisión de los datos hacia Internet.

Mediante un dashboard IoT, se podrá conocer en tiempo real la concentración del CO₂ así como la temperatura y la humedad del recinto donde el dispositivo esté colocado. Además, el sistema enviará una alerta mediante la app de mensajería Telegram cada vez que la concentración de CO₂ en el aire alcance un valor a partir del cual se considera que existe un alto riesgo de contagio.

PALABRAS CLAVE: IoT, CO₂, COVID-19, LPWAN, LoRaWAN, LoRa, TTN, Raspberry Pi

ABSTRACT

Technologies related to the Internet of Things are laying the foundations for an increasingly digitalised society. In particular, LPWANs (Low Power Wide Area Networks) are opening up a new horizon of possibilities where low power consumption and long range are paramount.

Due to the recent COVID-19 pandemic, the consequences of which we are still suffering today, this project aims to reduce the risk of contagion by measuring the concentration of CO₂ in indoor spaces. To this end, this work makes use of LoRaWAN technology for the interconnection of the nodes and data transmission to the Internet.

By means of an IoT dashboard, it will be possible to know in real time the CO₂ concentration as well as the temperature and humidity of the room where the device is placed. In addition, the system will send an alert via the Telegram messaging app every time the CO₂ concentration in the air reaches a value above which it is considered that there is a high risk of contagion.

KEYWORDS: IoT, CO₂, COVID-19, LPWAN, LoRaWAN, LoRa, TTN, Raspberry Pi

Índice del documento

1. Introducción.....	4
1.1. Motivación.....	4
1.2. Objetivos	5
1.3. Estructura de la memoria	6
2. Internet de las Cosas.....	7
2.1. Características de una red IoT	8
2.2. Arquitectura de un sistema IoT	9
2.3. Casos de uso	10
3. Relación existente entre la COVID-19 y el CO ₂	12
4. Estado del arte	16
4.1. Tecnologías de comunicación inalámbrica	16
4.1.1. Características de redes LPWA	17
4.1.2. Tecnologías LPWAN celulares.....	19
4.1.3. Tecnologías LPWAN no celulares.....	22
4.1.4. Comparativa entre tecnologías de comunicación LPWA	35
4.2. Placas de desarrollo.....	36
4.3. Sensores.....	42
4.3.1. Sensor de CO ₂ MH-Z19B	42
4.3.2. Sensor de CO ₂ MQ-135.....	45
4.3.3. Sensores de temperatura y humedad DHT11 y DHT22	48
4.4. Gateways	50

4.4.1.	Tipos de gateways	52
4.4.2.	RAK2245 Pi HAT + Raspberry Pi 4B.....	52
4.4.3.	MikroTik wAP LR8 kit	55
4.4.4.	Wirnet iStation.....	55
4.4.5.	LIG16 Indoor Gateway	56
4.4.6.	Comparativa entre gateways LoRaWAN	57
4.5.	Protocolos de comunicación IoT.....	58
4.5.1.	¿Qué es un protocolo?.....	58
4.5.2.	Características que debe cumplir un protocolo IoT.....	58
4.5.3.	AMQP	59
4.5.4.	CoAP	62
4.5.5.	MQTT.....	65
4.5.6.	Comparativa entre protocolos IoT	71
4.6.	The Things Network	72
4.7.	Plataformas IoT	75
4.7.1.	Datacake.....	75
4.7.2.	Ubidots.....	78
4.7.3.	ThingSpeak.....	81
4.7.4.	Node-RED	84
5.	Desarrollo de la solución	90
5.1.	Especificaciones y estructura de la solución.....	90
5.2.	Implementación de los nodos	91
5.2.1.	Componentes utilizados	91
5.2.2.	Organigrama	96
5.2.3.	Diseño de la carcasa 3D	97

5.3.	Implementación del gateway	100
5.3.1.	Puesta en marcha de la Raspberry Pi 4B y el concentrador.....	100
5.3.2	Diseño del prototipo 3D	104
5.4.	Servidor TTN.....	106
5.4.1.	Registro de los nodos LoRa en TTN	106
5.5.	Procesado de datos y plataformas IoT	114
5.5.1.	Procesado de datos	114
5.5.2.	Datacake.....	116
5.5.3.	Node-RED	122
6.	Resultados obtenidos	132
6.1.	Funcionamiento de los nodos.....	132
6.2.	Funcionamiento del dashboard IoT	134
6.3.	Funcionamiento de las alertas.....	136
7.	Presupuesto	138
8.	Conclusiones y líneas futuras.....	140
8.1.	Conclusiones.....	140
8.2.	Líneas futuras	141
9.	Índice de figuras	142
10.	Índice de tablas	148
11.	Referencias	149
12.	Glosario	160
	ANEXO I: Configuración del IDE de Arduino.....	162
	ANEXO II: Código Utilizado.....	164

1. Introducción

1.1. Motivación

El Internet de las Cosas (IoT) se ha convertido en uno de los sectores más prometedores en los últimos años. Este nuevo paradigma se basa en la interconexión de distintos sensores y dispositivos de muy diversa índole. Dispositivos como un reloj con conectividad Wi-Fi que mide los pasos de un paciente, un frigorífico inteligente que alerta de la caducidad de los productos o incluso un sensor ubicado en un parking que notifica cuando hay un coche aparcado, se incluyen dentro del ecosistema del IoT. Estos son solo algunos de los miles de objetos que pueden conectarse a Internet con el objetivo de interactuar entre sí sin la intervención humana. Hoy en día, prácticamente todos los sectores de la sociedad están siendo partícipes de la Cuarta Revolución Industrial, también conocida como Industria 4.0, basada en la digitalización y automatización de procesos que hasta ahora, eran gestionados por humanos (Murata et al., 2020). Algunos de ellos son el sector de la industria, el sector sanitario, el transporte, la banca o las telecomunicaciones (Roland Berger, 2016).

A raíz de la pandemia causada por la COVID-19, fuimos realmente testigos de los beneficios que nos aporta la tecnología. Algunos de ellos son el teletrabajo, permitiendo a muchas personas seguir trabajando desde casa y las clases online, las cuales han ayudado a muchos alumnos a continuar con sus estudios. Además, gracias a los recientes avances tecnológicos, también se han desarrollado sistemas y aplicaciones capaces de reducir el riesgo de contagio de la COVID-19. La medición de la temperatura corporal mediante un termómetro láser o la calidad del aire en entornos cerrados también han ayudado a gestionar la pandemia de una manera más efectiva, reduciendo así el número de contagios e identificando aquellos entornos mal ventilados.

De hecho, mantener una buena ventilación en espacios cerrados de forma regular puede ser tan efectivo como la vacunación del 50% de la población, ya que la mayor parte de los contagios se producen en entornos cerrados (Actility, 2021). Por este motivo, en este TFG se propone el desarrollo de un dispositivo que, haciendo uso de la tecnología IoT LoRaWAN, permita monitorizar los niveles de concentración de CO₂ y determinar así la calidad del aire.

1. Introducción

De esta forma, se pretende aprovechar las ventajas de las tecnologías IoT para disminuir el riesgo de contagio en todo tipo de recintos cerrados, mejorando de esta forma la calidad de vida de las personas que hagan uso de dichos recintos.

1.2. Objetivos

Como se ha comentado en el apartado anterior, el propósito de este trabajo fin de grado es el desarrollo y puesta en marcha de una serie de dispositivos IoT capaces de medir en tiempo real los niveles de concentración de CO₂ presente en el aire, así como los de la temperatura y humedad. Además, el sistema desarrollado deberá enviar una alerta al usuario cada vez que el CO₂ supere un determinado umbral. Asimismo, una plataforma IoT deberá mostrar de forma visual todas las medidas recogidas por los sensores mediante gráficas y barras de nivel. La plataforma IoT escogida como parte de la solución será accesible de manera remota a través de Internet.

De este modo, una vez detallado el enfoque de la solución propuesta, se van a exponer a continuación los diferentes objetivos propuestos para este proyecto:

- Desarrollar un sistema IoT formado por múltiples nodos que, haciendo uso de la tecnología LoRaWAN, sean capaces de medir el nivel de CO₂ así como la temperatura y humedad hacia un servidor.
- Desarrollar un sistema que permita la visualización de los datos recogidos por los sensores mediante un dashboard IoT.
- Crear un sistema que permita alertar a los usuarios cuando haya un riesgo alto de contagio.

1. Introducción

1.3. Estructura de la memoria

El presente proyecto se divide de la siguiente manera:

- En el apartado 1 *Introducción*, se argumentará en general sobre la revolución que supone la irrupción de las tecnologías IoT en la sociedad así como los objetivos que se han de cumplir para desarrollar un sistema que permita disminuir el riesgo de propagación en interiores a causa de la COVID-19.
- En el apartado 2 *Internet de las Cosas*, se explicará el concepto de Internet de las Cosas así como sus características y los casos de uso de este nuevo paradigma.
- En el apartado 3 *Relación existente entre la COVID-19 y CO₂*, se analizará en detalle el motivo por el que el riesgo de contagio aumenta cuando hay una gran concentración de CO₂ en el aire.
- En el apartado 4 *Estado del arte*, se analizarán las tecnologías y los dispositivos que se implementarán en el proyecto así como otros de características similares y se justificará la elección de cada uno.
- En el apartado 5 *Desarrollo de la solución*, se explicará paso a paso el funcionamiento de cada sensor elegido así como la puesta en marcha del servidor TTN, la creación del dashboard IoT Datacake y además, se explicará cómo se desarrolla un bot interactivo.
- En el apartado 6 *Resultado obtenidos*, se analizará el funcionamiento de los sensores así como el funcionamiento del servidor TTN y las alertas de los niveles de CO₂.
- En el apartado 7 *Presupuesto*, se desglosará cada uno de los costes de este trabajo.
- Finalmente, en el apartado 8 *Conclusiones*, se comentarán las conclusiones tras el desarrollo del proyecto así como las posibles mejoras del mismo.

2. Internet de las Cosas

El IoT o Internet of Things es un nuevo paradigma tecnológico que hace referencia a la interconexión de dispositivos tanto físicos como virtuales con el objetivo de transmitir distintos tipos de datos hacia Internet para su posterior procesado. Generalmente, es posible controlar de forma remota y en tiempo real tanto la información transmitida por los sensores como el estado de los mismos.

El término anglosajón *things* traducido como “cosas”, hace referencia a cualquier dispositivo electrónico que posee un microprocesador capaz de gestionar y procesar la información recogida por los sensores y enviarla, a continuación, a través de Internet, ya sea directa o indirectamente mediante un gateway. Algunos ejemplos de dispositivos que se podrían incluir dentro de una red “IoT” serían una pulsera cuantificadora capaz de medir las pulsaciones de una persona y, a su vez, enviarlas a un servidor mediante una conexión Wi-Fi o un sensor que permita compartir la ubicación de un animal en tiempo real mediante la tecnología Sigfox.

Si bien el desarrollo del IoT se ha acelerado en los últimos años, la primera vez que se utilizó el término “Internet de las cosas” fue en el año 1999. Concretamente, Kevin Ashton, emprendedor e investigador británico, fue quien acuñó este término durante una presentación para la compañía Procter & Gamble's (Elder, 2019). Desde entonces, el número de dispositivos y desarrollos relacionados con el IoT ha aumentado exponencialmente. De hecho, tal y como se puede apreciar en la Figura 1, desde el cuarto trimestre del año 2020, el número de conexiones IoT superó, por primera vez, al de conexiones no IoT. Esto significa que desde ese momento, las conexiones entre máquinas y sensores superan a aquellas que realizan los usuarios con sus teléfonos inteligentes y ordenadores (Pelaez, 2020).

2. Internet de las Cosas

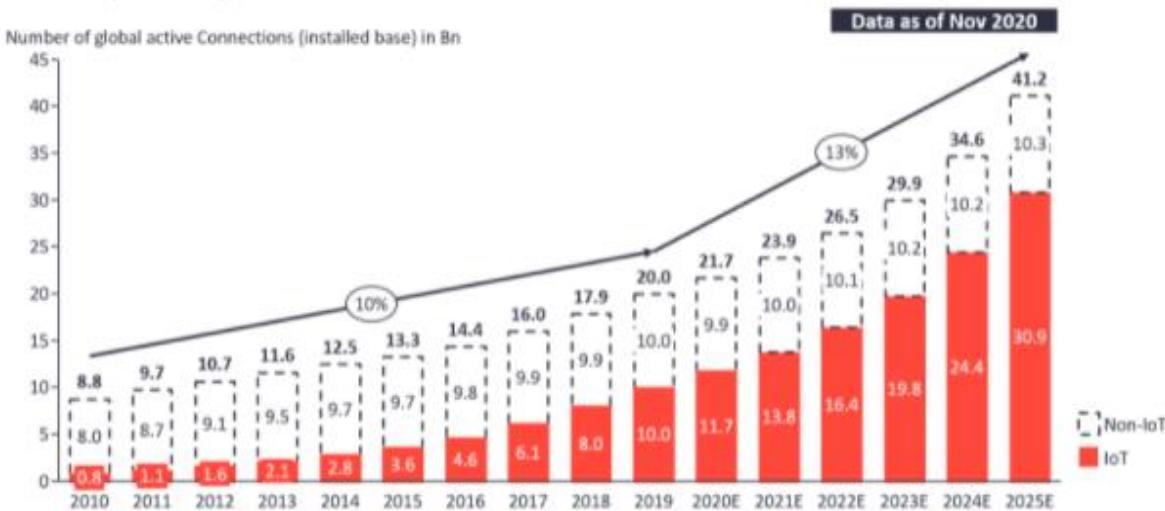


Figura 1. Conexiones totales de dispositivos inteligentes desde 2010 (Fuente: IoT Analytics)

2.1. Características de una red IoT

Aunque hoy en día existen multitud de redes IoT con distintas tecnologías, todas ellas poseen características similares, destacando las siguientes (Hostingplus, 2020):

- **Sensorización.** Los dispositivos IoT se utilizan principalmente para la monitorización y medición de parámetros de todo tipo como la calidad del aire, la detección de movimiento, la velocidad, la luminosidad, entre otros.
- **Interacción completa entre personas y máquinas.** Si bien las redes IoT se asemejan a las redes máquina a máquina (M2M), también deben facilitar la información transmitida a los usuarios mediante interfaces o dashboard.
- **Seguridad.** Debido al aumento en el número de ataques informáticos a redes IoT, la seguridad es un parámetro que toda empresa que desarrolla y diseña redes de sensores inteligentes debe tomar en consideración.

- **Escalabilidad y fácilmente adaptable.** Ya se trate de un sensor de movimiento o uno que transmita la temperatura cada media hora, existen diferentes protocolos que se adaptan a cada situación. Por ejemplo, si se desea transmitir datos con un gran ancho de banda a una distancia menor a 20 metros, el Wi-Fi será la mejor opción. Si por el contrario se desea transmitir únicamente la humedad de una granja cuya ubicación es una zona rural aislada, una tecnología LPWA será la opción más conveniente debido al bajo ancho de banda de los mensajes y al gran rango de cobertura que ofrece.
- **Mínima capacidad de procesamiento.** Los dispositivos IoT deben contar con una mínima capacidad de procesamiento para poder procesar los datos procedentes de los sensores y enviarlos hacia un gateway o router.

2.2. Arquitectura de un sistema IoT

Toda arquitectura IoT, independientemente de la tecnología utilizada y del escenario donde se despliegue, comprende principalmente 5 elementos que se describen a continuación (AlfaIOT, s.f.):

- **Dispositivos hardware.** Hace referencia a los sensores que hacen mediciones de parámetros físicos.
- **Software.** Se trata del microcontrolador ya que este es capaz de procesar la información procedente de los sensores gracias a su microprocesador.
- **Comunicaciones.** Esta parte engloba a los gateways y puntos de acceso, los cuales son capaces de receptionar las señales digitales procedentes de los microcontroladores y reenviarlas hacia Internet.
- **Plataformas en la nube.** Esta parte se refiere a todas aquellas plataformas que permiten almacenar la información en bases de datos así como su visualización mediante diferentes widgets y herramientas.

2. Internet de las Cosas

- **Aplicaciones en la nube.** Se trata de la última parte de una arquitectura IoT y se incluyen aquellas aplicaciones que permiten procesar toda la información mediante algoritmos con Machine Learning e Inteligencia Artificial (IA) para obtener datos precisos. Esta parte despierta un gran interés entre las empresas ya que cuantos más datos se recopilen y se analicen, mejores decisiones se podrán tomar.

La Figura 2 recoge de manera esquemática todas las partes que forman una arquitectura IoT:

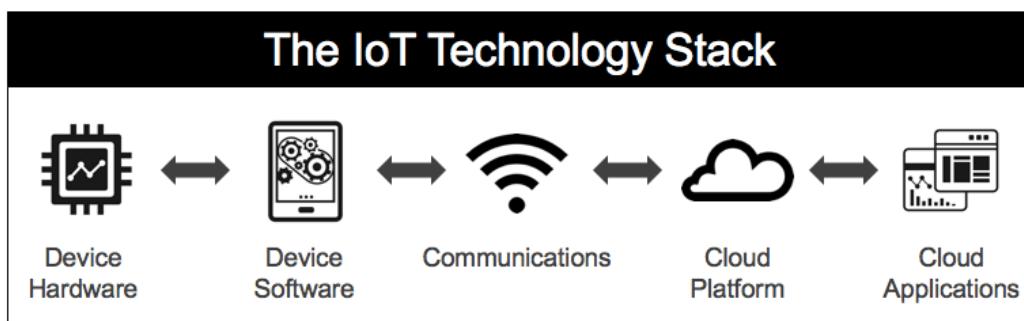


Figura 2. Arquitectura de un sistema IoT dividido en 5 capas (Fuente: AlfaIOT)

2.3. Casos de uso

El IoT se puede desplegar en escenarios de muy diversa índole, entre los cuales se pueden encontrar los siguientes:

- **Industria.** Gracias al IoT, se puede monitorizar la ubicación de un producto, el estado del mismo así como optimizar los procesos industriales. Por ejemplo, rastrear en tiempo real la ubicación exacta de los contenedores que viajan en un barco para evitar su pérdida o monitorizar el funcionamiento de una máquina en tiempo real para que, en caso de que ocurriese cualquier contratiempo, fuese posible identificar el fallo rápidamente. El Internet de las Cosas que se emplea en la industria se denomina IIoT (Industrial Internet of Things) (Posey, Rosencrance y Shea, 2021).

2. Internet de las Cosas

- **Domótica.** Otro sector que ha sufrido un gran avance en los últimos años es la domótica. La apertura de puertas desde el teléfono móvil, frigoríficos que avisan cuando un producto va a caducar o luces que se encienden con la voz mediante Alexa son sólo algunos ejemplos de todas las aplicaciones que el IoT aporta a los hogares.
- **Ciudades inteligentes.** Además de la domótica y la industria, el potencial del IoT puede ser aprovechado en las ciudades, dando lugar al término Smart Cities o ciudades inteligentes. Algunos ejemplos serían el despliegue de sensores en carreteras para limitar la velocidad máxima en función del estado de las carreteras, sensores en los parkings para informar a los conductores de las plazas libres o la medición de la calidad del aire.
- **Medicina.** Gracias al avance del IoT, hoy en día es posible insertar un dispositivo inteligente en el corazón de aquellos pacientes que padeczan de este con el objetivo de monitorizar las constantes vitales de forma remota. Los relojes inteligentes o wearables son otro claro ejemplo de dispositivos inteligentes capaces de medir la cantidad de pasos, la frecuencia cardíaca o la cantidad de horas que una persona duerme. Todos estos aparatos conectados a Internet permiten recopilar diferentes datos y enviarlos a una aplicación o base de datos para, posteriormente, hacer un seguimiento más preciso del paciente (Bigelow, 2014).

3. Relación existente entre la COVID-19 y el CO₂

3. Relación existente entre la COVID-19 y el CO₂

Antes de analizar qué relación existe entre la COVID-19 y el CO₂, es fundamental entender qué es la COVID-19 así como los síntomas que provoca en los humanos. La COVID-19 es una enfermedad causada por SARS-CoV-2, un nuevo coronavirus que surgió en la ciudad china de Wuhan a finales del 2019. El 30 de enero de 2020, la OMS declaró que la COVID-19 se trataba de una emergencia de salud pública a nivel internacional, y el 11 de marzo de ese mismo año, la caracterizó como una pandemia (Organización Panamericana de la Salud, 2020). Entre los síntomas que manifiestan las personas contagiadas destacan la tos seca, la fiebre, la dificultad para respirar y, en menor medida, también puede provocar la pérdida del olfato o del gusto, dolor de cabeza o incluso dificultad para respirar (disnea) (Organización Mundial de la Salud, 2020). El SARS-CoV-2 se transmite por 3 vías distintas (Departamento de Salud Ambiental de Madrid, 2020), la primera de ellas por la vía respiratoria, la segundo por el contacto con superficies y zonas contaminadas y la tercera, vía fecal-oral.

No obstante, tal y como se puede apreciar en la Figura 3, el tiempo que permanece el virus en el aire en forma de aerosol es muy superior en comparación a otras superficies como el plástico, el cobre o el acero inoxidable. Por tanto, la vía de transmisión más contagiosa es la respiratoria.

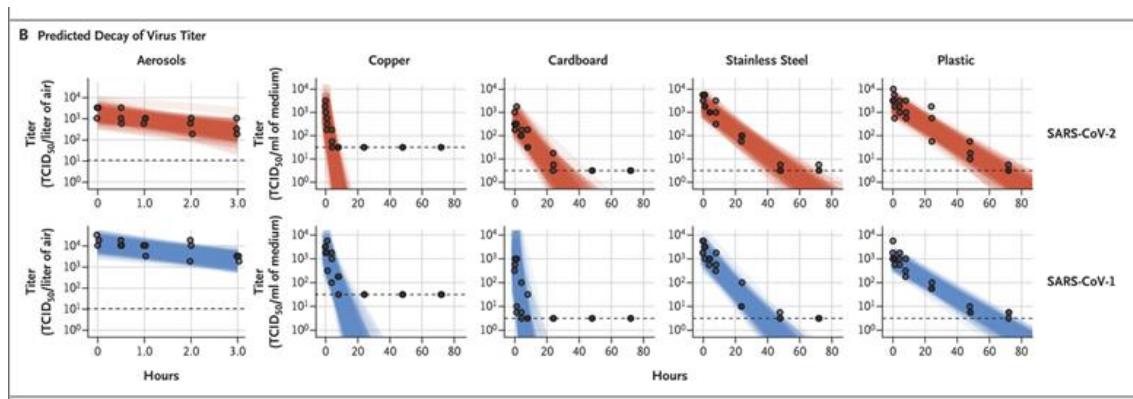


Figura 3. Concentración de partículas del virus SARS-CoV-2 en diferentes escenarios (van Doremale et al., 2020)

3. Relación existente entre la COVID-19 y el CO₂

Cada vez que una persona infectada habla, estornuda, tose o simplemente exhala aire, también expulsa diminutas gotas que se dividen en 2 grupos. Por un lado se encuentran las gotículas, las cuales son aquellas partículas cuyo tamaño es superior a $300\mu m$, mientras que aquellas que no superan los $100\mu m$, se denominan aerosoles (Zafra y Salas, 2020). Estas últimas quedan suspendidas en el aire durante un cierto tiempo. No obstante, los aerosoles formados por partículas con un tamaño inferior a $5\mu m$ pueden quedar suspendidas en el aire durante 3 horas (van Doremalen et al., 2020). Si posteriormente otra persona respira el mismo aire contaminado, puede llegar a infectarse sobre todo si se encuentra en una zona cerrada, ya que la probabilidad de contagio en un recinto cerrado con mala ventilación es entre 20 y 100 veces superior a la de una zona al aire libre (COIIAR, 2021).

La forma más efectiva de reducir los contagios en zonas cerradas, además de llevar la mascarilla, es mediante una ventilación adecuada. Existen 3 tipos de ventilación (COIIAR, 2021):

- **Ventilación mecánica:** La renovación del aire se produce mediante un aparato electromecánico el cual introduce aire procedente del exterior y filtra el que hay en el interior.
- **Ventilación natural:** Simplemente consiste en abrir las ventanas y las puertas del recinto para que se renueve el aire.
- **Ventilación mixta:** Se trata de una combinación de ambos tipos.

No obstante, para determinar cuál es el mejor momento para ventilar un recinto, debemos conocer previamente la calidad del aire que se respira en él. Para ello, se mide la concentración de CO₂. Este gas no sólo se produce cuando tiene lugar la combustión de combustibles fósiles, sino también cuando exhalamos aire. Una elevada concentración de CO₂ en una sala cerrada puede indicar la existencia de una gran cantidad de partículas presentes en el aire, es decir, de aerosoles.

3. Relación existente entre la COVID-19 y el CO₂

El riesgo aumenta cuando una de las personas presentes está contagiada de COVID-19, ya que los aerosoles producidos por esta contienen partículas infectadas que se esparcen por todo el recinto y, a su vez, otras personas volverán a respirar. En caso de que la ventilación sea escasa o muy pobre, el riesgo de contagio aumenta ya que el aire no se renueva. Por este motivo, es fundamental una buena ventilación en entornos cerrados (García Hernández, 2021).

Según el RITE (Reglamento de Instalaciones Térmicas en los Edificios), existen 4 niveles de CO₂ aceptables en función del tipo de local, los cuales se recogen en la siguiente tabla:

Categoría (Calidad del aire interior)	Descripción	Concentración aceptable	Concentración límite
IDA 1 (Calidad de aire óptima)	Hospitales, clínicas, laboratorios.	350 ppm	750 ppm
IDA 2 (Buena calidad de aire)	Oficinas, museos, centros educativos, residencias	500 ppm	900 ppm
IDA 3 (Calidad media de aire)	Centros comerciales, hoteles, restaurantes	800 ppm	1200 ppm
IDA 4 (Baja calidad de aire)	No se debe aplicar	1200 ppm	-

Tabla 1. Categorías del aire interior en función del tipo de local (Fuente: Asociación Técnica Española de Climatización y Refrigeración)

3. Relación existente entre la COVID-19 y el CO₂

Sin embargo, en una situación de pandemia, la concentración de CO₂ en cualquier tipo de local o edificio no debe sobrepasar los 700 ppm ya que, a partir de este valor, el riesgo de contagio se incrementa (COIIAR, 2021).

En el presente trabajo, se considerará un nivel aceptable de CO₂ aquellos valores que no superen los 700 ppm. A partir de este valor, es aconsejable ventilar el aula y, en el supuesto en el que la concentración de CO₂ supere los 1000 ppm, habría que ventilar el aula con urgencia (Axiomet, s.f.).



Figura 4. Nivel de riesgo en función de la concentración de CO₂ (Fuente: Axiomet)

4. Estado del arte

En este apartado se van a analizar las distintas tecnologías, protocolos, plataformas y dispositivos IoT que se van a utilizar para desarrollar la solución planteada así como otras similares. Además, se justificará la elección de cada uno de los elementos utilizados en este trabajo.

4.1. Tecnologías de comunicación inalámbrica

Al diseñar un proyecto IoT, uno de los aspectos más importantes que debemos tener en cuenta es la tecnología de comunicación que permite la conexión de los nodos a Internet. Cada proyecto posee unas características distintas y, por tanto, no todas las tecnologías son válidas en todos los escenarios.

Tecnologías como Wi-Fi, Bluetooth o incluso ZigBee permiten la transmisión de datos a altas velocidades junto con un gran ancho de banda. No obstante, el alcance máximo de este tipo de redes no supera los 100 metros. Estas características son idóneas en entornos domésticos o en el interior de edificios (Pothuganti & Chitneni, 2014). No obstante, existen otro tipo de redes que ofrecen una cobertura muy superior con un bajo consumo energético, a costa de sacrificar ancho de banda. Estas redes se denominan LPWAN (*Low Power Wide Area Network*) o redes de largo alcance y bajo consumo.

Las principales tecnologías LPWAN son: LTE-M, NB-IoT, LoRaWAN y Sigfox. Las dos primeras pertenecen al grupo de redes “celulares” mientras que las dos últimas son “no celulares”.

A continuación, se detallarán cada una de las tecnologías mencionadas anteriormente junto con sus características así como su funcionamiento. Además, se realizará una comparativa para ver las principales diferencias de cada una.

4. Estado del arte

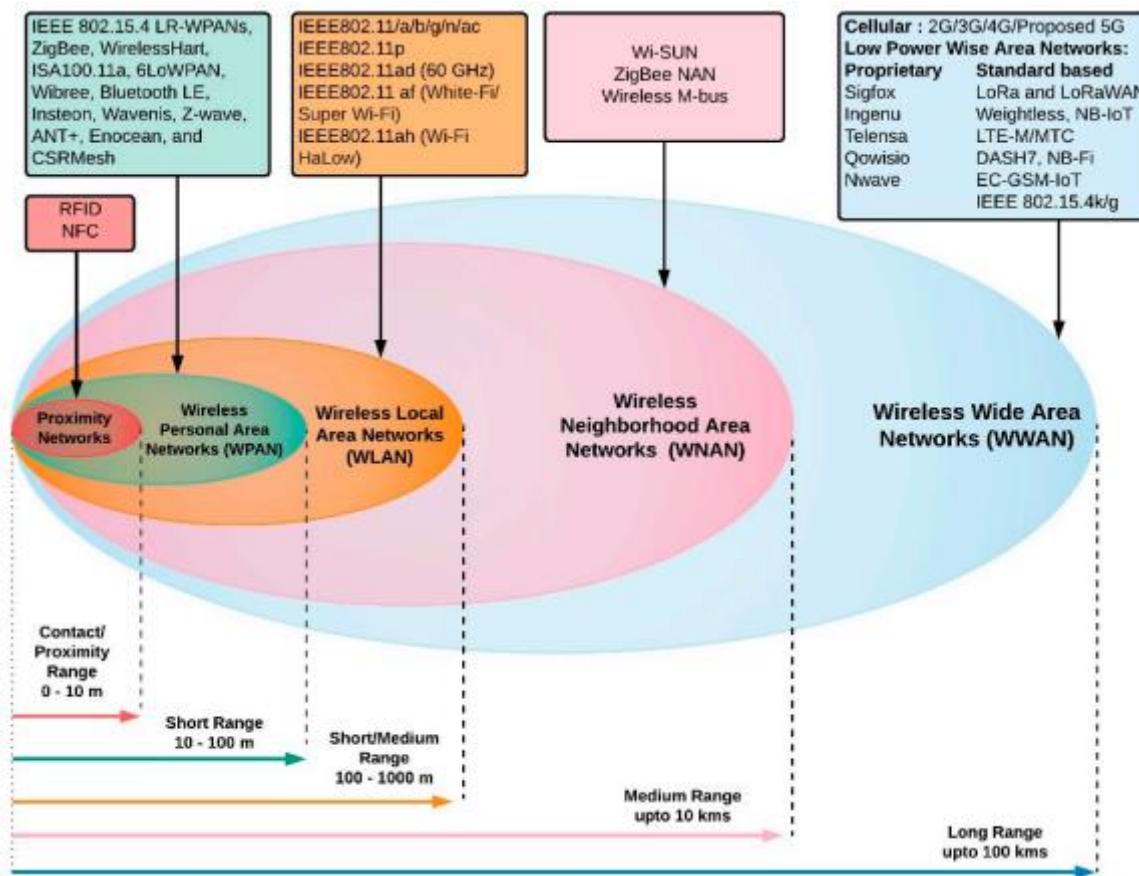


Figura 5. Comparativa entre las tecnologías de comunicación inalámbrica (Chaudhari, Zenaro y Borkar, 2020)

4.1.1. Características de redes LPWA

En multitud de ocasiones nos solemos plantear siempre la misma cuestión: ¿Cuál es la mejor tecnología que podemos implementar en nuestro proyecto IoT? La realidad es que no existe la tecnología perfecta ya que todo depende de las especificaciones de cada proyecto. No obstante, todas las aplicaciones y dispositivos que utilicen una tecnología LPWA, deben cumplir las siguientes características:

4. Estado del arte

- **Bajo consumo:** Un bajo ancho de banda junto con una baja frecuencia (si se compara con otras tecnologías celulares) permiten aumentar la vida útil de los dispositivos hasta 10 años con una sola carga, dependiendo de la tecnología empleada.
- **Bajo coste:** Para favorecer la adopción de nuevas soluciones IoT, los costes deben ser muy reducidos con precios en descenso a medida que se aplica economía de escala.
- **Escalabilidad y compatibilidad con otras tecnologías:** Las infraestructuras IoT deben ser fácilmente escalables debido al progresivo aumento de nuevos dispositivos ya que cada vez hay un mayor número de dispositivos y, por tanto, es necesario que los nuevos sensores que se vayan añadiendo a la red sean compatibles con los ya existentes.
- **Soporte para multitud de dispositivos:** El número de sensores IoT crece de manera exponencial cada año. Es por ello que las redes e infraestructuras deben estar preparadas para soportar y gestionar de manera simultánea un gran número de sensores y dispositivos sin que otros factores se vean afectados, como el ancho de banda, la latencia o la eficiencia energética.
- **Amplia cobertura:** Uno de los rasgos diferenciadores de las redes LPWA con respecto a otras tecnologías es su gran alcance. El hecho de utilizar frecuencias por debajo de los GHz (consultar frecuencias de LTE-M) permite aumentar su rango de cobertura además de tener una gran penetración en interiores, bajo tierra y en entornos subacuáticos (Chaudhari, Zenaro y Borkar, 2020).

4.1.2. Tecnologías LPWAN celulares

➤ LTE-M / eMTC

LTE-M (Long Term Evolution for Machines), también conocida como LTE Cat-M1 o eMTC, es una tecnología celular introducida por el organismo 3GPP en la release 13 en 2015. Comparte la misma red que el LTE aunque está diseñada para la comunicación entre dispositivos IoT.

Esta tecnología LPWA hereda gran parte de las ventajas del LTE, como una transmisión de datos cifrada, el envío de SMS o la posibilidad de transmitir voz gracias a VoLTE (*Voice over LTE*). No obstante, ha sido diseñada principalmente para comunicaciones Máquina a Máquina (M2M), permitiendo una transmisión a baja potencia además de ser energéticamente eficiente. La batería de los dispositivos puede durar entre 5 y 10 años gracias a la implementación de dos modos de ahorro energético, el PSM (*Power Saving Management*) y el eDRX (*extended Discontinuous Reception*). La combinación de ambos permite que el dispositivo esté siempre en reposo y únicamente se despierte cuando va a transmitir información. La cobertura es mayor en comparación al LTE convencional, superando los 11 km en algunas situaciones.

Opera en las mismas bandas que el LTE aunque su ancho de banda se reduce hasta 1.08 MHz. La velocidad máxima de transmisión es de 300 kbps en bajada y 380 kbps en subida (aunque el máximo teórico es 1 Mbps). La latencia está entre 100 y 150 ms. Los modos de duplexación compatibles son tanto FDD (*Frequency Division Duplexing*) como TDD (*Time Division Duplexing*) y la potencia de transmisión máxima es de 23 dBm (Nokia Networks, 2015).



Figura 6. Logo comercial de LTE-M (Fuente: GSMA)

4. Estado del arte

➤ NB-IoT

NB-IoT (*Narrowband Internet of Things*), también llamado LTE Cat-NB1, es una tecnología LPWA celular introducida por la 3GPP en la release 13 en 2016, y surge como una mejora al LTE-M. Mientras que esta opera únicamente en las bandas del LTE convencional, NB-IoT puede implementarse en distintas partes del espectro de GSM (2G) y LTE (4G). Las 3 principales zonas del espectro donde se implementa son:

- **In-band:** Puede compartir bandas de frecuencia con el LTE.
- **Guard band:** Puede operar en las bandas de guarda del LTE. Estas zonas del espectro no se utilizan ya que están destinadas a evitar interferencias entre dos transmisiones cercanas.
- **Standalone operation:** Puede reutilizar las bandas de frecuencia de GSM.

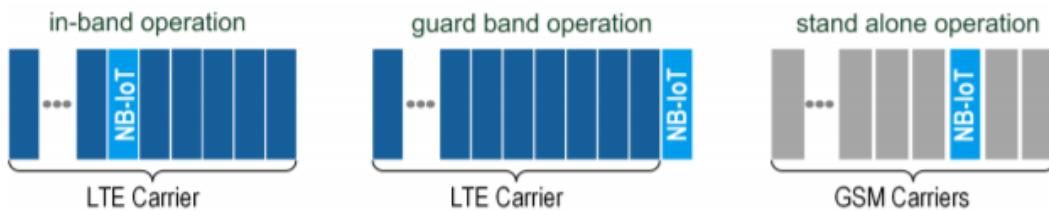


Figura 7. Bandas en las que opera NB-IoT (Fuente: Rohde Schwarz)

Con respecto a las características técnicas, utiliza un ancho de banda de 200 kHz cuando opera en las bandas de GSM, mientras que en bandas de LTE su ancho de banda es de 180 kHz. Su velocidad de transmisión es menor en comparación con las de LTE-M, siendo 250 kbps en bajada y 20 kbps en subida, y la latencia se sitúa en torno a 1 segundo.

Estas características permiten a los dispositivos aumentar su vida útil, hasta los 10 años, así como alcanzar un rango de cobertura superior a 15 km en determinadas circunstancias (Foubert & Mitton, 2020). No obstante, no es posible desarrollar esta tecnología en aplicaciones

4. Estado del arte

que requieran de movilidad o deseen transmitir voz. Únicamente soporta el modo de duplexación FDD aunque al igual que LTE-M, los modos de ahorro energético eDRX y PSM también son compatibles y permiten alargar la batería de los dispositivos hasta 10 años. La potencia de transmisión máxima de 23 dBm (T-Mobile, 2019).



Figura 8. Logo comercial de NB-IoT (Fuente: GSMA)

➤ Comparativa entre LTE-M y NB-IoT

A grandes rasgos, las principales diferencias entre el LTE-M y NB-IoT, son:

- **LTE-M:** Aplicaciones IoT que requieren de un gran ancho de banda, movilidad y transmisión de voz.
- **NB-IoT:** Soluciones que requieren de una baja tasa de transmisión de datos, sensores estáticos y buena penetración en interiores.

La figura 5, muestra un mapa con los países en donde se implementa cada una de las tecnologías descritas anteriormente, donde el color rojo corresponde con LTE-M, el azul con NB-IoT, mientras que el color morado corresponde a los países donde están desplegadas ambas tecnologías:

4. Estado del arte

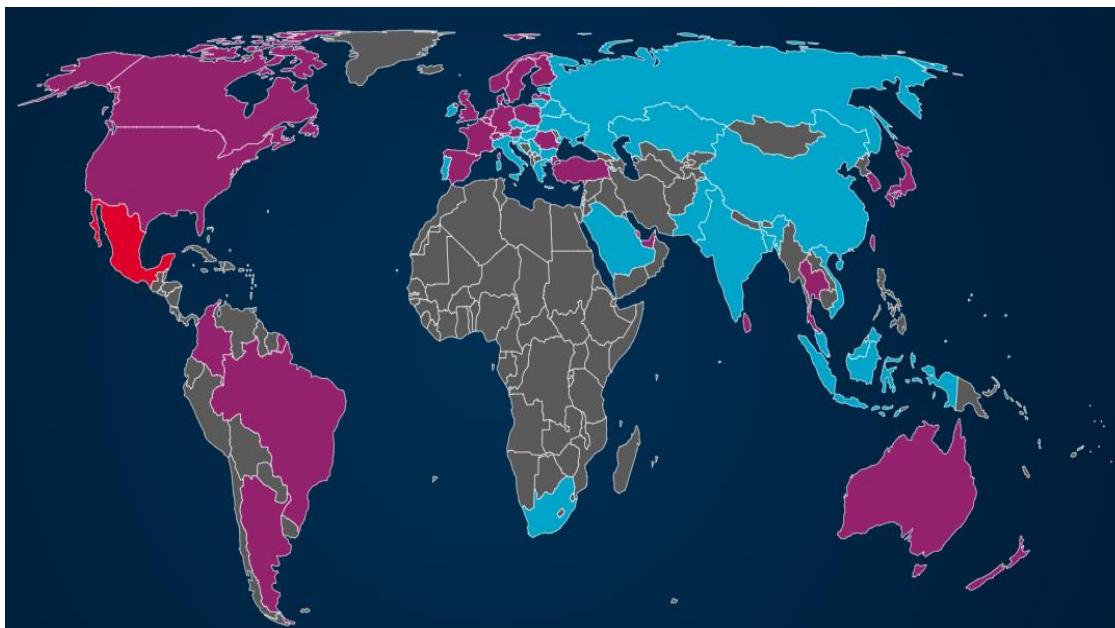


Figura 9. Mapa de cobertura de LTE-M y NB-IoT (Fuente: GSMA)

4.1.3. Tecnologías LPWAN no celulares

A diferencia de las tecnologías LPWAN celulares, las LPWAN no celulares suelen ser tecnologías propietarias y su principal característica se basa en las bandas en las que operan, ISM (*Industriales, Científicas y Médicas*). Para explotar este tipo de bandas no se requiere de ningún tipo de licencia ya que estas están destinadas a un uso no comercial en sectores industriales, científicos y médicos.

Antes de analizar el funcionamiento así como las características de esta tecnología, conviene aclarar que LoRa y LoRaWAN no son términos semejantes. El término de LoRa hace referencia a la capa física (PHY) del protocolo así como al tipo de modulación, ambos patentados por la empresa francesa Semtech. Por otra parte, LoRaWAN define la capa de control de acceso al medio (MAC) que posibilita la transmisión de los datos. La Figura 10 muestra las distintas capas de protocolos que forman la tecnología de LoRa así como las partes que las integran.

4. Estado del arte

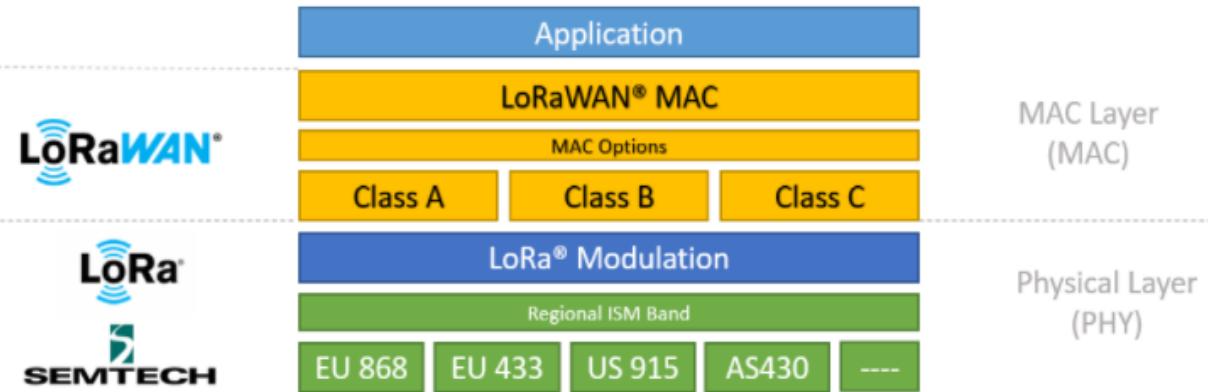


Figura 10. Pila de protocolos de LoRaWAN (Fuente: LoRa Alliance)

➤ LoRaWAN

LoRaWAN es un protocolo de comunicaciones regulado por LoRa Alliance que define la capa de control de acceso al medio (MAC, capa 2) y se basa en una topología de estrella. Su función principal es coordinar y gestionar los datos enviados por los nodos LoRa a Internet, así como el control de diversos parámetros como la velocidad de transmisión, el cifrado de mensajes, la vida útil de los nodos o la potencia de transmisión.

La arquitectura de LoRaWAN está formada principalmente por 4 elementos (The LoRa Alliance, s.f.):

- 1) **Nodos LoRa:** Normalmente los nodos cuentan con uno o varios sensores encargados de recopilar y procesar distintos parámetros físicos como pueden ser la temperatura, la humedad, la vibración, el nivel de agua, el nivel de Co₂, o la luminosidad, entre otros. Toda esta información es transmitida hacia los gateways mediante la modulación LoRa en señales de radio. Además de los sensores, otro tipo de dispositivos muy utilizados por su gran utilidad son los actuadores. Estos son capaces de ejecutar distintas acciones al recibir una señal eléctrica como por ejemplo, encender y apagar una bombilla, o abrir y cerrar una válvula.

4. Estado del arte

- 2) **Gateways o puertas de enlace:** Su función principal consiste en recibir los datos procedentes de los nodos y enviarlos al servidor de red de LoRaWAN a través del protocolo TCP/IP. No existe una conexión fija entre los gateways y los nodos, es decir, un nodo puede enviar la misma señal a varios gateways simultáneamente, mejorando así la robustez de la comunicación, aumentando la autonomía de los dispositivos y disminuyendo la PER (tasa de error de paquetes o en inglés, *packet error rate*).
- 3) **Servidor de red:** Varias puertas de enlace se conectan al servidor de red mediante una conexión TCP/IP de forma inalámbrica o con cable Ethernet. Su misión es regular la velocidad de transmisión mediante un mecanismo llamado ADR (Adaptive Data Rate), realizar comprobaciones de seguridad y filtrar los paquetes duplicados.
- 4) **Servidor de aplicación:** Este tipo de servidores se encargan de almacenar los datos recibidos en bases de datos para posteriormente mostrarlos en diferentes plataformas y aplicaciones IoT. Algunas aplicaciones ya incluyen tanto la base de datos como el dashboard para visualizar todos los datos recibidos, como por ejemplo Ubidots.

La Figura 11 muestra de forma gráfica todas las partes anteriormente descritas de la red LoRaWAN:

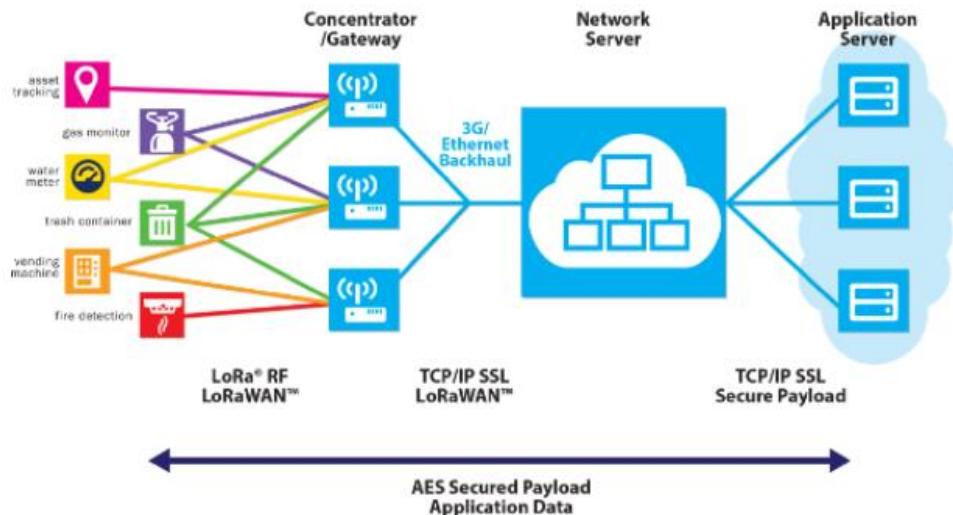


Figura 11. Estructura de la red LoRaWAN (Fuente: LoRa Alliance)

4. Estado del arte

Como se puede apreciar en la Figura 11, LoRaWAN se basa en una topología en estrella que a su vez depende de otra estrella más grande. Es decir, existen multitud de gateways que se conectan a un servidor y, a su vez, cientos de dispositivos están conectados a cada uno de los gateways.

❖ Seguridad en LoRaWAN

Si bien la seguridad no es un elemento como tal, sí está presente de forma intrínseca pues es fundamental proteger y cifrar toda la información transmitida. La red LoRaWAN hace uso del cifrado dinámico AES-128 para la transmisión de los datos. Se pueden distinguir dos capas de seguridad en la red LoRaWAN:

- Capa de red **TCP/IP SSL LoRaWAN**: Hace uso de una clave AES de 128 bits llamada clave de sesión de red (NwkSKey) para asegurar los datos transmitidos en la red.
- Capa de aplicación **TCP/IP SSL Secure Payload**: Hace uso de una clave AES de 128 bits llamada clave de sesión de aplicación (AppSKey) que permite cifrar de extremo a extremo toda la información para que el operador de red no tenga acceso a los datos (Canvision Systems, 2021).

❖ Clases de dispositivos

En función del tipo de comunicación utilizada, la latencia y el ahorro energético, los dispositivos LoRa se dividen en 3 clases:

- **Clase A**: En esta clase se encuentran los dispositivos que suelen utilizar una batería como fuente de alimentación. Se trata de la más eficiente pues únicamente permite la comunicación *downlink* (de bajada) cuando el nodo LoRa haya enviado un mensaje *uplink* (*de subida*).

4. Estado del arte

- **Clase B:** Esta clase es similar a la Clase A pero el gateway puede transmitir mensajes en determinados momentos sin esperar a que el nodo haya enviado un mensaje *uplink*. Esto es posible gracias a una sincronización entre los dispositivos y el gateway.
- **Clase C:** Esta es la clase menos eficiente ya que los nodos LoRa están permanentemente escuchando y por tanto el gateway puede transmitir mensajes *downlink* en cualquier momento (Sabas, 2017).

➤ LoRa

LoRa es un tipo de modulación basada en la técnica de espectro ensanchado Chirp Spread Spectrum (CSS). Diseñada en los años 40 para radares y usos militares, esta técnica consiste en transmitir señales digitales en forma de *chirps*, unas señales sinusoidales en el dominio de la frecuencia que pueden aumentar o disminuir en el tiempo. Cuando el gateway recibe estas señales, este las demodula y las convierte en bits.

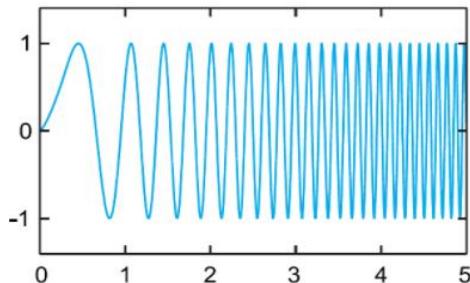


Figura 12. Chirp de espectro ensanchado (Fuente: Semtech)

Con respecto a las características técnicas de LoRa, cabe destacar que el ancho de banda se puede configurar de 3 posibles formas, 125, 250 o 500 kHz y su máxima velocidad de transmisión es de 50 kbps de bajada y 0,3 kbps de subida. Cuenta con una alta sensibilidad de -168dB, lo que permite una buena recepción de datos. La vida útil de los dispositivos alcanza puede superar los 10 años en determinadas situaciones. La banda de frecuencias en las que

4. Estado del arte

opera LoRa son las de 868 MHz en Europa, 915 MHz en América y 433 MHz en Asia, mientras que su rango de cobertura está comprendido entre 10 y 15 km, en función de la zona donde se despliegue. Este amplio rango de cobertura permite abastecer a un extenso número de dispositivos con un solo gateway (Sabas, 2017).

La técnica CSS está basada en la modulación DSSS (*Direct Sequence Spread Spectrum*), que consiste en variar la fase de la señal portadora de acuerdo con una secuencia definida por un código. Esto se consigue mediante el producto entre este código, llamado *spreading code* (o código de dispersión) y la señal que contiene la información a transmitir. En la Figura 13 se puede ver el proceso de modulación.

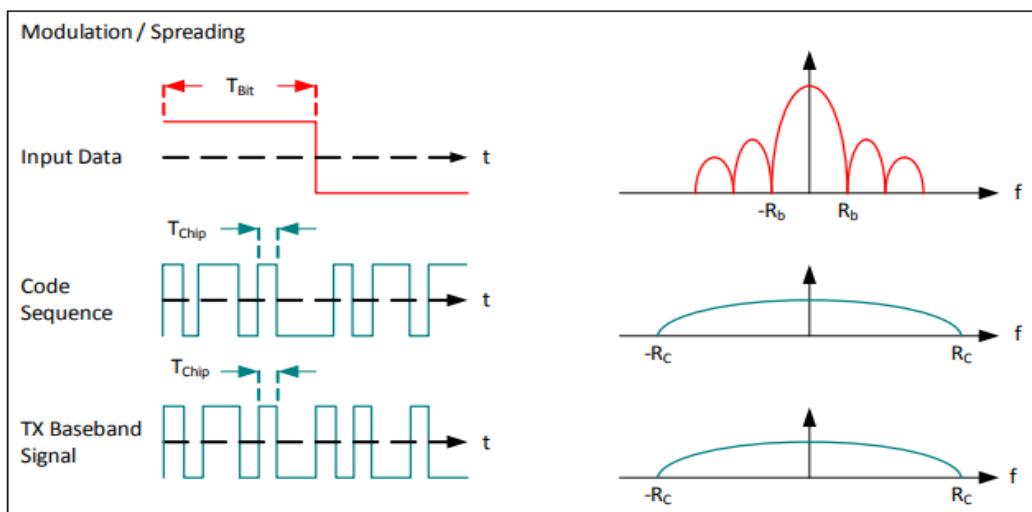


Figura 13. Modulación DSSS (Fuente: Solera, 2018)

Respecto al proceso de demodulación, es similar al de modulación ya que basta con multiplicar la señal recibida por el mismo código de dispersión que se usó en la modulación.

Mediante esta modulación, se puede obtener una mayor ganancia en la transmisión de los datos, incluso cuando la relación señal a ruido (SNR) es negativa. No obstante, a pesar de las ventajas que ofrece, la modulación DSSS requiere de un reloj de gran precisión y muy bien sincronizado. Este tipo de características hacen que la recepción de los mensajes sea un proceso

4. Estado del arte

complejo, sobre todo si se pretende ahorrar la máxima energía posible, ya que los dispositivos deben estar constantemente “escuchando” por si llegan nuevos mensajes.

Ante esta circunstancia, la compañía Semtech patentó la modulación LoRa. Esta solventaba los principales problemas de la modulación DSSS, convirtiéndola en una alternativa robusta, de bajo consumo y bajo coste. En la modulación LoRa, la dispersión del espectro es posible gracias a la generación de un chirp, que varía en el dominio de la frecuencia constantemente.

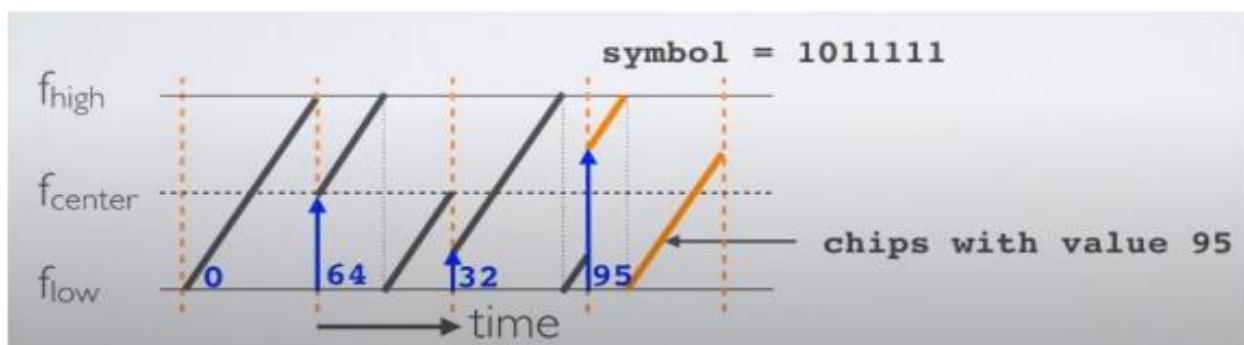


Figura 14. Señal modulada mediante DSSS (Mobilefish, 2019)

Como se puede apreciar en la Figura 14, cada tiempo de símbolo (T_s), en la figura aparece como *time*, la frecuencia parte de un determinado valor, comprendido entre una frecuencia de inicio y una frecuencia final. El valor desde el cual parte la frecuencia es el símbolo del mensaje que se envía. El número de bits que es posible codificar viene determinado por el parámetro SF (Spreading Factor o factor de ensanchamiento), el cual define 2 parámetros distintos, por un lado define el número de bits que forman un símbolo, y por otro las combinaciones totales de un símbolo. Cabe resaltar que cada símbolo puede contener un máximo de 2^{SF} chips o valores de frecuencia a los que puede “saltar” (Mobilefish, 2019).

Por ejemplo, si tuviésemos un factor de ensanchamiento $SF = 7$, tendríamos por un lado símbolos formados por 7 bits y por otro lado 128 chips posibles, ya que $2^7 = 128$. El SF, junto con el ancho de banda, se utilizan también para hallar la tasa de envío de símbolo (Rs) (velocidad), así como el tiempo de símbolo (T_s).

4. Estado del arte

$$T_s = \frac{2^{SF}}{B} \quad (1)$$

$$R_s = \frac{B}{2^{SF}} \quad (2)$$

donde T_s es el tiempo de símbolo, B es el ancho de banda y R_s es la tasa de envío de símbolo.

De las ecuaciones (1) y (2) se puede observar que, al aumentar el ancho de banda B , disminuye el tiempo de símbolo T_s y por tanto aumenta la velocidad o tasa de envío de símbolo R_s . Además, también se puede deducir que al aumentar el parámetro SF , aumenta el tiempo de símbolo T_s y por ende, la velocidad disminuye, aunque mayor inmunidad frente a interferencias y mayor alcance tendrá la comunicación. Asimismo, cuanto mayor sea el factor de ensanchamiento, menor será la potencia a la que el receptor puede recibir la señal (menor sensibilidad), pero a cambio la comunicación será más robusta.

Existen un total de 6 factores de ensanchamiento, desde SF7 hasta SF12, que se eligen automáticamente en función de varios parámetros como la distancia, la SNR o la RSSI. En la siguiente tabla se puede apreciar cómo varía la tasa de bit y la distancia en función del SF elegido:

Factor de ensanchamiento (SF)	Tasa de bit	Distancia (depende del terreno)	Sensibilidad del receptor
SF10	980 bps	8 km	-132 dBm
SF9	1760 bps	6 km	-130 dBm
SF8	3125 bps	4 km	-127 dBm
SF7	5470 bps	2 km	-125 dBm

Tabla 2. Relación del factor de ensanchamiento con otros parámetros (Fuente: LoRaWAN)

4. Estado del arte

Otra característica importante de LoRa es que las señales con un SF distinto son ortogonales entre sí. Esto permite que aquellas señales con la misma frecuencia y ancho de banda no se interfieran entre ellas. Asimismo, las comunicaciones de LoRa son inmunes a efectos negativos tales como la propagación multicamino, el desvanecimiento o el efecto Doppler.

❖ **Métodos de autenticación**

LoRaWAN ofrece 2 modos de autenticación para registrar los dispositivos en la red:

- **OTAA** (Over-the-Air Activation): Se trata del método más seguro ya que cada nueva conexión, las claves de sesión de Aplicación (AppSKey) y sesión de Red (NwkSKey) se borran y se generan otras nuevas. Previo a la transmisión de información, el dispositivo debe llevar a cabo un procedimiento de unión mediante la personalización de este. Para ello, varios parámetros deben ser configurados (Rivas, 2021):
 - **DevEUI**: Se trata de una identificación que posee cada dispositivo de forma única. Con cierto parecido a la MAC, este identificador utiliza las direcciones IEEE EUI64.
 - **AppEUI**: Es un identificador que se almacena en la memoria del dispositivo antes de que este se active. Permite identificar el servidor de aplicación y se asemeja al número de puerto. Al igual que DevEUI, utiliza direcciones IEEE EUI64.
 - **AppKey**: A diferencia de los parámetros anteriores, AppKey es una clave de 128 bits que permite generar las claves de sesión NwkSKey y AppSKey. Tanto el servidor como el dispositivo tienen acceso a esta clave.

Una vez realizado el procedimiento de unión, el dispositivo envía un mensaje para unirse a la red, el cual contiene el AppEUI, AppKey y DevEUI. Si el servidor acepta la solicitud de unión, este responde mediante un mensaje con la siguiente información:

- **DevAddr**: Se trata de un número de 32 bits que permite identificar al dispositivo dentro de la red. Se asemeja a una dirección IP.
- **NetID**: Se trata de un identificador de red.

4. Estado del arte

- **AppNonce:** El servidor proporciona al dispositivo este valor para generar las claves NwkSKey y AppSKey.
- **ABP** (Activation by Personalization): Este método es el más sencillo pues no se lleva a cabo el proceso de solicitud de unión. Esto es posible ya que tanto el DevAddr como NwkSKey y AppSKey se almacenan en el dispositivo, pudiendo así conectarse a la red sin necesidad de configurar el DevEUI, AppEUI o AppKey. Es decir, el dispositivo ya lleva registrados los parámetros necesarios para unirse a la red (Rivas, 2021).

En este trabajo se utilizará el método de activación OTAA ya que, si bien ABP es un proceso más sencillo, no ofrece tanta seguridad como sí lo hace OTAA.

➤ SigFox

Sigfox es una tecnología no celular de largo alcance y bajo consumo desarrollada por la compañía francesa que lleva el mismo nombre. Se trata de la primera operadora dedicada exclusivamente a ofrecer una red de comunicación IoT por todo el mundo, estando presente en 72 países a día de hoy (Redacción Data Center Market, 2021). A diferencia de la red LoRaWAN donde son los usuarios quienes instalan sus propios gateways para ampliar la red, en este caso es SigFox quien se encarga de instalar y desplegar sus antenas y servidores. Para la transmisión de información, SigFox emplea dos modulaciones de banda ultra-estrecha (UNB) llamadas DBPSK (Differential Binary Phase Shift Keying) y GFSK (Gaussian Frequency-Shift Keying). La primera de ella se utiliza en la comunicación ascendente mientras que la segunda para la descendente. El hecho de utilizar un ancho de banda de 100 Hz, la velocidad máxima está limitada a 600 bps de bajada y 100 bps de subida.

Las características descritas anteriormente permiten alcanzar una cobertura de 30 a 50 km en entornos rurales mientras que en entornos urbanos la cobertura varía entre 3 y 10 km. En Europa, Sigfox opera en la banda de 868 MHz, en EEUU lo hace a 902 MHz y Asia a 433 MHz.

4. Estado del arte

Con respecto a la batería de los dispositivos, esta puede superar los 10 años (Aernouts, Berkvens, Van Vlaenderen, & Weyn, 2018).

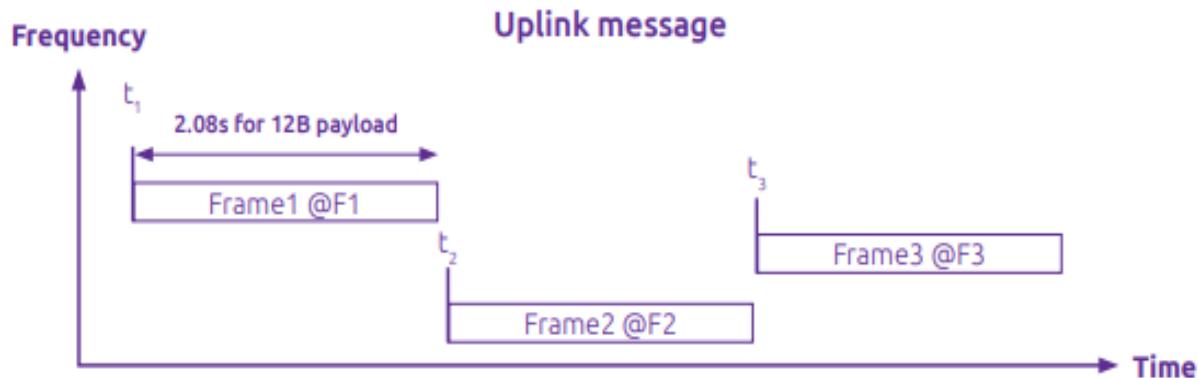


Figura 15. Mensaje con sus réplicas (Fuente: Sigfox)

Para aumentar la calidad de la comunicación, SigFox emplea una técnica llamada Random Access o acceso aleatorio que consiste en que cada mensaje se envía junto con 2 réplicas empleando 3 frecuencias y 3 tiempos distintos, tal y como muestra la Figura 15. Además, se trata de una comunicación asíncrona entre el dispositivo final y el servidor.

La red de Sigfox cuenta con algunas limitaciones relacionadas con la cantidad de mensajes que se pueden enviar a diario así como con su tamaño. Con respecto a la transmisión ascendente, se pueden enviar hasta 140 mensajes diarios de 12 bytes cada uno. En enlace descendente, el máximo es de 4 mensajes al día de 8 bytes cada uno.

4. Estado del arte

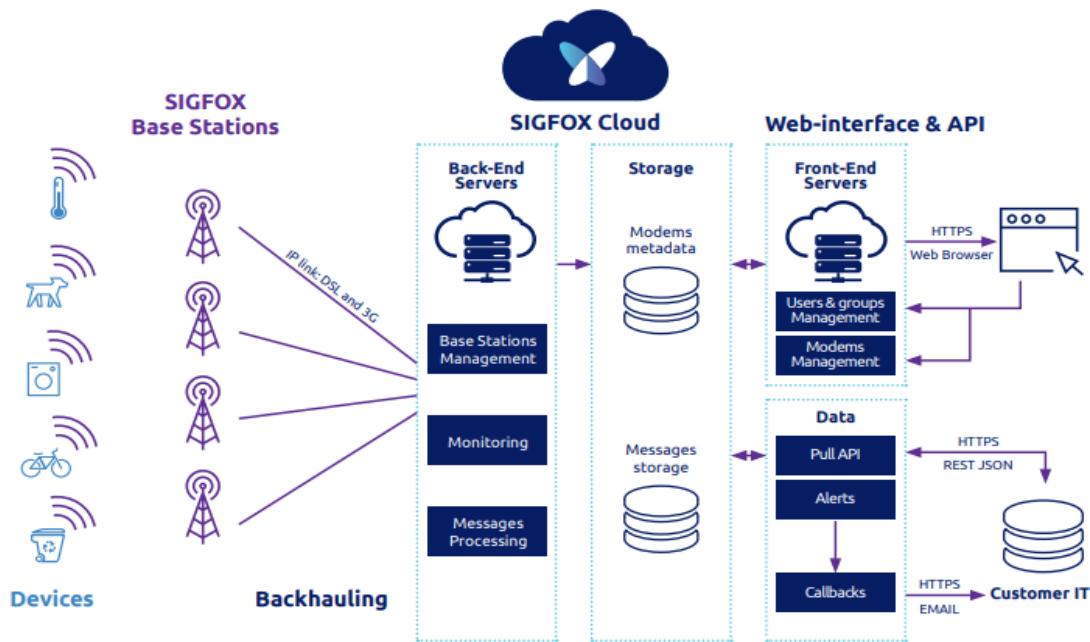


Figura 16. Arquitectura SigFox (Fuente: Sigfox)

La arquitectura de SigFox, similar a la de LoRaWAN, está formada por los dispositivos finales, las estaciones base, la SigFox Cloud la cual contiene los servidores back-end que gestionan y almacenan los datos, y los servidores front-end que gestionan la interfaz web así como la API para visualizar los datos en plataformas de terceros. Todos estos elementos están representados de manera gráfica en la Figura 16.

De forma resumida, el proceso para transmitir información en una red SigFox sería el siguiente:

- 1) Los nodos envían mensajes hacia las estaciones base.
- 2) Las estaciones base reenvían los mensajes hacia la SigFox Cloud haciendo uso de la conectividad 3G/4G o en su defecto, mediante una conexión vía satélite.
- 3) La SigFox Cloud gestiona el estado de la red y de las estaciones base. Además, comprueba si se han recibido varios mensajes idénticos, y almacena aquellos que son válidos.

4. Estado del arte

4) Todos los mensajes se envían hacia una web llamada SigFox Backend donde los usuarios pueden visualizar los datos. Además, existe la posibilidad de redirigir todos los datos recibidos hacia una base de datos o aplicación IoT, mediante dos opciones que ofrece Sigfox:

- Por un lado, Sigfox pone a disposición de los usuarios una API, cuyo funcionamiento se basa en HTTP REST que devuelve los mensajes en formato JSON.
- Por otro lado, podemos hacer uso de una opción llamada callback, que consiste en indicar al back-end de Sigfox la página web donde queremos recibir los datos. De esta forma, cada vez que llegue un mensaje al servidor de Sigfox, este será reenviado a la URL indicada.

❖ Seguridad en SigFox

SigFox ofrece 3 niveles distintos de seguridad que pueden implementarse en los dispositivos:

- **Nivel medio:** Las credenciales de seguridad se almacenan en la memoria del dispositivo.
- **Nivel alto:** Las credenciales de seguridad se almacenan en una zona segura basada en software.
- **Nivel muy alto:** Las credenciales de seguridad se almacenan en un elemento seguro que permite encriptar la información transmitida mediante el sistema AES de cifrado (SigFox, 2017).

4.1.4. Comparativa entre tecnologías de comunicación LPWA

Tecnología	LTE-M	NB-IoT	LoRa	SigFox
Espectro	Licencia	Licencia	Sin licencia	Sin licencia
Ancho de banda	1,4 MHz	180-200 KHz	125-25-500 KHz	100 Hz
Banda frecuencia	700 - 900 MHz	700 - 900 MHz	430-868-915 MHz	433-868-902 MHz
Modulación	16QAM	QPSK	CSS (LoRa)	DBPSK-GFSK
Sentido comunicación	Half Duplex Full Duplex	Half Duplex	Half Duplex	Half Duplex
Velocidad	1 Mbps 20 kbps (UL)	250 kbps (DL) 20 kbps (UL)	50 kbps (DL) 0,3 kbps (UL)	600 bps (DL) 100 bps (UL)
Rango	11 km	15 km	15 km (rural) 5 km (urbano)	50 km (rural) 10 km (urbano)
Vida útil dispositivos	5 - 10 años	10 años	10 años	10 años

Tabla 3. Comparativa entre tecnologías de comunicación LPWA (Foubert y Mitton, 2019)

Una vez detalladas las 4 principales tecnologías de comunicación LPWA junto con sus características, es momento de explicar cuál se ha escogido para este trabajo.

Las tecnologías celulares (LTE-M y NB-IoT) se han descartado ya que suponen un mayor coste económico pues para poder utilizar este tipo de tecnologías hay que pagar una tarifa mensual, además de tener un mayor consumo energético si se compara con las tecnologías no celulares. Bien es cierto que las tecnologías celulares ofrecen una mayor velocidad, sin embargo,

4. Estado del arte

los parámetros que se desean transmitir en este trabajo como son la temperatura, el CO₂ y la humedad no requieren grandes tasas de transmisión.

Por tanto, la elección quedaría entre SigFox y LoRa. Aunque a priori estas dos tecnologías puedan parecer similares, existen algunas diferencias que hacen que LoRa sea la tecnología más idónea para este proyecto:

- La posibilidad de instalar una red propia. Es decir, a diferencia de SigFox, LoRaWAN permite que cualquier usuario pueda instalar su propio gateway para ofrecer cobertura LoRa allá donde no haya sin depender de ninguna empresa. Se trata por tanto de una red de comunicaciones descentralizada.
- SigFox es una tecnología que todavía no está desplegada en todos los países del mundo y por tanto existen zonas en las que no es posible utilizarla todavía.
- Baja dificultad de aprendizaje ya que en Internet existen numerosos tutoriales y una extensa documentación sobre la tecnología de LoRa y LoRaWAN.
- Apoyar un proyecto IoT open-source para que más usuarios puedan disfrutar de una red de comunicaciones IoT gratuita.

4.2. Placas de desarrollo

Las placas de desarrollo son dispositivos electrónicos con un microcontrolador integrado capaz de ejecutar distintas instrucciones para un fin concreto. Estas placas cuentan con interfaces de entrada y salida a los que se le conectan distintos sensores, tanto analógicos como digitales. Los datos recogidos por estos son procesados por el microcontrolador.

En un sistema IoT, los microcontroladores son una de las partes más importantes ya que sin ellas, los sensores carecerían de utilidad y no sería posible obtener la información deseada. Diversos factores han favorecido el aumento exponencial de estos dispositivos en proyectos y soluciones IoT, pero sin duda los más destacados son su facilidad de uso y su precio, el cual es

4. Estado del arte

cada vez más económico. El sector energético, sanitario, industrial o doméstico son algunos donde la adopción del IoT se ha incrementado en los últimos años (Placas de Desarrollo, s.f.). Desde la creación de Arduino en el año 2003, han surgido multitud de placas con distintas características, diseñadas para todo tipo de aplicaciones y tecnologías (Fernández, 2020). Por este motivo, es fundamental elegir la placa que mejor se adapte a nuestro proyecto, teniendo en cuenta sus características técnicas así como la tecnología que utiliza para transmitir datos.

Antes de describir las distintas placas de desarrollo más utilizadas a día de hoy, es de suma importancia introducir el ESP32, el SoC más utilizado a día de hoy junto con el ESP8266.

➤ ESP32

El ESP32 es un SoC de bajo consumo energético y de bajo coste para aplicaciones IoT fabricado por la empresa Espressif. Posee un procesador Tensilica Xtensa LX6 de 32 bits de doble núcleo y trabaja a una frecuencia de 160 MHz, aunque es posible aumentarla hasta los 240 MHz.

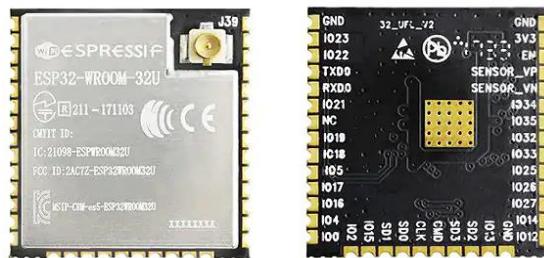


Figura 17. Módulo ESP32-WROOM-32U (Fuente: Espressif)

Respecto al hardware, la Tabla 4 recoge las especificaciones técnicas más importantes:

4. Estado del arte

Especificaciones ESP32	Descripción
Procesador	Tensilica Xtensa LX6 32 bits a 160 - 240 MHz
Wi-Fi	802.11 b/g/n 2.4 GHz
Bluetooth	v4.2 / BLE
Memoria RAM	520 KB
Memoria Flash	16 MB
Pines GPIO	22 pines
Conversor ADC (analógico a digital)	18 pines
Conversor DAC (digital a analógico)	2 pines
Interfaces	UART, I2C, SPI

Tabla 4. Especificaciones del ESP32-WROOM-32U (Fuente: Nabto)

En cuanto al software, los lenguajes de programación más utilizados para programar el ESP32 son C/C++ y MicroPython. No obstante, también existen otros lenguajes de programación compatibles como RTOS, Mongoose OS, Lua o Espruino (Hübschmann, 2020). En función de los sensores que queramos emplear, debemos escoger la librería más adecuada para cada uno.

Entre todas las placas de desarrollo que llevan el ESP32 integrado, las más utilizadas son:

4. Estado del arte

- **ESP32-DevKitC:** Se trata de una placa de desarrollo básica desarrollada por Espressif para iniciarse en el mundo del IoT. Posee el módulo ESP32-WROOM-32D dual core. Cuenta con una memoria Flash de 4MB y una RAM de 8 MB. Incorpora varios LEDs así como 2 botones (Hübschmann, 2020).



Figura 18. ESP32-DevKitC (Fuente: Nabto)

- **NodeMCU-32S:** Esta placa de desarrollo diseñada por AI Thinker es muy parecida al ESP32-DevKitC aunque a diferencia de este, el NodeMCU-32S posee el módulo ESP32-WROOM-32 single core (Hübschmann, 2020).

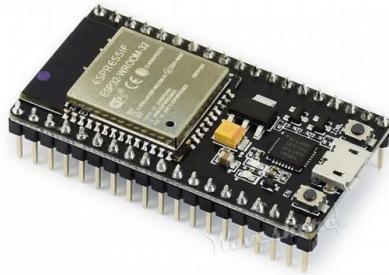


Figura 19. Node-MCU-32S (Fuente: Nabto)

- **ESP32-CAM:** A diferencia del resto de placas, esta destaca por poseer una cámara integrada así como un slot para introducir una tarjeta microSD. Su módulo es un ESP32-S y cuenta con 9 pines únicamente. El modelo de la cámara es OV2640 y posee una resolución de 2 Mpx. El único inconveniente es la carencia de una interfaz micro USB, por lo que este dispositivo se carga mediante un adaptador FTDI (Hübschmann, 2020).

4. Estado del arte



Figura 20. *ESP32-CAM* (Fuente: Espressif)

- **Heltec Wi-Fi LoRa 32 V2.1:** Se trata de un chip que lleva integrado un módulo ESP23 y un chip LoRa SX1276, fabricado por Semtech. Además, incorpora una pantalla OLED de 0.96 pulgadas, un convertidor USB-serial CP2102 y un conector SH1.25-2 para conectar una batería externa. Incorpora una antena Wi-Fi helicoidal y un conector para una antena externa SMA (antena LoRa). Cuenta con 22 pines GPIO digitales (entrada y salida). Se alimenta a 3,3V pero soporta hasta 6V. La ganancia de la antena de LoRa es de 2 dBi mientras que la sensibilidad del receptor es de -139 dBm. Su consumo en Deep Sleep es de 800µA (Heltec Automation, s.f.).

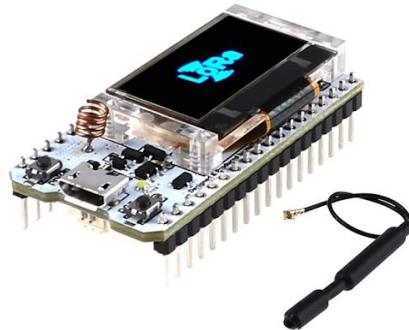


Figura 21. *Heltec Wi-Fi LoRa 32 V2.1* (Fuente: Heltec)

4. Estado del arte

- **TTGO T-Beam V2.1:** Al igual que el Heltec Wi-Fi LoRa 32 V2.1, este dispositivo incorpora un módulo de ESP32 PICO-D4 y el mismo chip LoRa, el SX1276. Cuenta además con un módulo GPS NEO-6M y el voltaje de entrada varía desde los 1,8V a 3,7V. Tiene 17 pines GPIO entrada/salida, una pantalla OLED de 0.96 pulgadas. Destaca su bajo consumo en Deep Sleep, únicamente 0.2µA. La antena Wi-Fi está impresa en la placa y cuenta con un conector IPX / IPEX para antena SMA (antena LoRa). El receptor posee la misma sensibilidad, -139 dBm. Destaca también la interfaz JST XH2 para baterías externas, un convertidor USB-serial CP2104, y una ranura para insertar tarjetas microSD (Unit Electronics, s.f.).



Figura 22. TTGO T-Beam V2.1 (Fuente: UE Electronics)

La placa de desarrollo que se ha elegido para este trabajo es la Heltec Wi-Fi LoRa 32 V2.1. Se han descartado todas aquellas que no son compatibles con LoRa pues es la tecnología de comunicación que se va a utilizar. Las diferencias con la placa TTGO T-Beam V2.1 son mínimas pues ambas poseen el mismo chip LoRa y unas especificaciones similares. No obstante, la placa de Heltec ocupa menos espacio ya que cuenta con unas dimensiones menores, mientras que el GPS es una funcionalidad que no se va a utilizar en este TFG. Además, la placa TTGO tiene un mayor coste frente a la placa de Heltec. Todas estas razones justifican la elección de la placa Heltec Wi-Fi LoRa 32 V2.1.

4.3. Sensores

Actualmente existen multitud de sensores que nos permiten medir el CO₂, la temperatura y la humedad. Es posible encontrar sensores cuyos precios y características pueden ser muy dispares, aunque un bajo precio no siempre implica una mala calidad. A continuación, se detallarán las características de los sensores más utilizados a día de hoy para medir distintos parámetros que pueden ser de gran utilidad para nuestro proyecto.

4.3.1. Sensor de CO₂ MH-Z19B

El MH-Z19B es un sensor fabricado por la compañía china Winsen que permite medir el dióxido de carbono (CO₂) que se encuentra en el aire haciendo uso del principio de infrarrojos no dispersivos (Non Dispersive Infrared Detector o NDIR). Este tipo de sensores se utiliza en multitud de entornos, como en los sistemas de ventilación, medición de la calidad del aire, domótica o en sistemas HVAC (Heating, Ventilating and Air Conditioning) (Winsen, s.f.).



Figura 23. MH-Z19B (Fuente: Winsen)

Este sensor se caracteriza por tener una alta sensibilidad, compensación de temperatura integrada así como un filtro de vapor de agua para evitar que se contamine. No obstante, la característica más destacable es la detección de CO₂ mediante el principio de infrarrojos no dispersivos (NDIR), que se basa en el principio de absorción de energía a una longitud de onda determinada.

4. Estado del arte

Todos aquellos gases que poseen 2 o más átomos diferentes como por ejemplo el dióxido de carbono (CO₂), metano (CH₄), dióxido de azufre (SO₂) o monóxido de carbono (CO) absorben radiación infrarroja (IR) a una longitud de onda distinta. El CO₂ absorbe radiación infrarroja entre los 14 μm y 16 μm (Academia Testo, 2018), como se puede apreciar en la Figura 24.

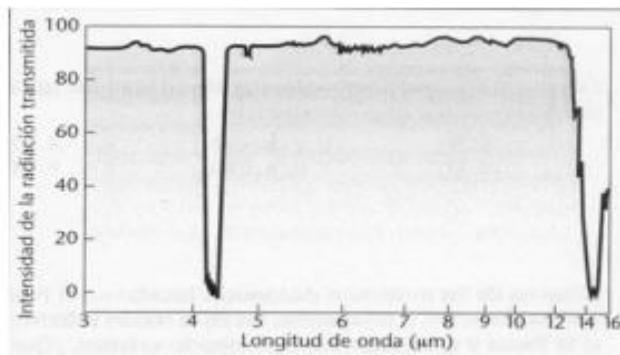


Figura 24. Espectro de absorción infrarroja del CO₂ (Fuente: UCO)

Este sensor en concreto se compone de un radiador de luz infrarroja, una cámara de medición, un filtro de absorción de CO₂ y un detector de radiación infrarroja. Para poder medir el CO₂ presente en el aire, la luz infrarroja viaja desde el emisor de luz hasta el detector IR a través de la cámara de medición donde está el gas. Teniendo en cuenta que en la atmósfera hay distintos tipos de gases y únicamente se desea medir el CO₂, el filtro empleado sólo deja pasar aquella parte del espectro de luz infrarroja que absorbe este gas (Academia Testo, 2018).

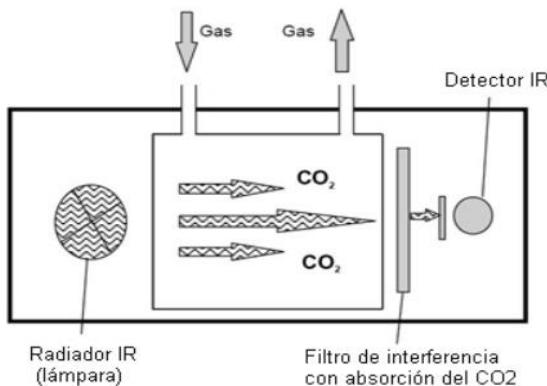


Figura 25. Parte interna del sensor MH-Z19B (Fuente: Academia Testo)

4. Estado del arte

Las principales características técnicas del sensor aparecen en la siguiente tabla:

Características	Descripción
Voltaje	4,5 V a 5,5 V
Corriente máxima	150 mA
Rango de medición	0 a 5000 ppm (partes por millón)
Señal de salida	PWM, TTL y UART
Tiempo de precalentamiento	3 min
Temperatura de funcionamiento	0º a 50º C
Humedad soportada	0% a 90%
Precisión	±75 ppm
Vida útil	5 años aprox.
Dimensiones	33 x 20 x 9 mm

Tabla 5. Especificaciones del MH-Z19B (Fuente: Winsen)

La concentración del CO₂ es directamente proporcional a la cantidad de radiación IR absorbida. La absorción viene determinada por la ley de Lambert-Beer (Wika, 2020):

$$A = -\log\left(\frac{\Phi}{\Phi_0}\right) = \varepsilon \cdot c \cdot l \quad (3)$$

4. Estado del arte

donde A es la absorción, Φ es la intensidad luminosa tras la absorción del gas, Φ_0 es la intensidad luminosa antes de la absorción, \mathcal{E} es el coeficiente de extinción, c es la concentración y l es la longitud de la cámara de medición.

4.3.2. Sensor de CO₂ MQ-135

El MQ-135 es un sensor electroquímico desarrollado por la compañía búlgara Olimex que permite medir diferentes tipos de gases, entre los que se encuentran el dióxido de carbono (CO₂), el benceno (C₆H₆), el amoniaco (NH₃), el humo y el óxido de nitrógeno (NO_x), entre otros gases. Tiene un rango de medición de 10 a 10000 ppm (Iberotecno, 2021).



Figura 26. MQ-135 (Fuente: UE Electronics)

El MQ-135 está protegido por una malla para evitar que cualquier tipo de partícula pueda dañar la sensibilidad del sensor. Además, antes de usarse, se recomienda realizar un proceso llamado *burning* (quemado en inglés) durante las primeras 24 horas, en el que cualquier partícula externa se quema mediante un calefactor que se encuentra en el interior del sensor.

Las especificaciones técnicas del sensor MQ-135 aparecen en la siguiente tabla.

4. Estado del arte

Características	Descripción
Voltaje	5 V
Corriente máxima	150 mA
Rango de medición	0 a 10000 ppm (partes por millón)
Señal de salida	TTL
Temperatura de funcionamiento	-20º a 70º C
Humedad soportada	0% a 95%
Dimensiones	32 x 22 x 24 mm

Tabla 6. Especificaciones del MQ-135 (Fuente: Olimex)

La Figura 27 muestra cómo varía la sensibilidad del sensor en función de la concentración de gases en el ambiente. El fabricante únicamente muestra los datos en un rango que va desde 10 a 110 ppm, aproximadamente. Las líneas representan cada uno de los gases y la separación entre ellas determina la capacidad del sensor de distinguir unos gases de otros. A priori parece que evolucionan de forma lineal, sin embargo, a medida que aumentan las partículas por millón, la trayectoria de cada gas empieza a variar, llegando incluso a mezclarse unos con otros. Esto es el margen de error del sensor (Iberotecno, 2021).

4. Estado del arte

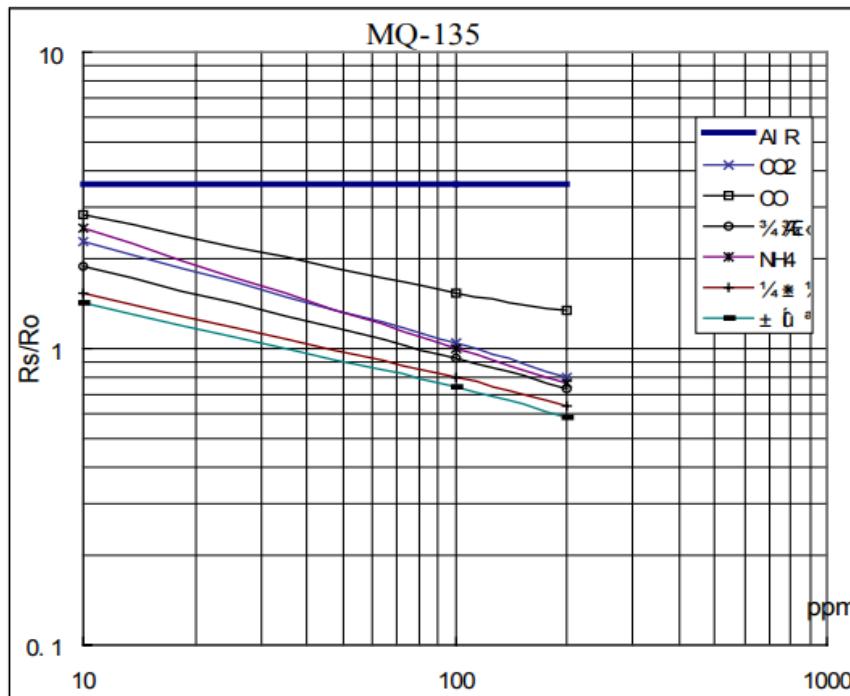


Figura 27. Sensibilidad del sensor MQ-135 (Fuente: Olimex)

Como se puede apreciar, cuando hay una baja concentración de gases, el sensor proporciona mediciones fiables, no obstante, una vez que se incrementa la concentración (a partir de 30 ppm aproximadamente), el margen de error aumenta y por ende, los resultados se vuelven imprecisos. Además, resulta energéticamente ineficiente debido a su alto consumo, en torno a 800 mW. Por tanto, a pesar de que el sensor MQ-135 puede medir una gran variedad de gases, no es de gran utilidad para este trabajo, pues la finalidad de este es obtener mediciones precisas de CO₂.

Por este motivo, el sensor que vamos a emplear para medir el CO₂ presente en el ambiente será el MH-Z19B, pues ofrece unas mediciones con mayor precisión.

4.3.3. Sensores de temperatura y humedad DHT11 y DHT22

En este punto se van a detallar las características de ambos sensores y se analizará cuál es el que mejor encaja en este proyecto:

- **Sensor DHT11:** El DHT11 es un sensor capaz de medir tanto la temperatura como la humedad relativa del aire. Su reducido precio, en torno a 6€, hace que sea uno de los sensores más utilizados en multitud de proyectos IoT. Lleva incorporada una resistencia pull-up de 5 kΩ así como un LED que se enciende cuando el sensor está encendido.

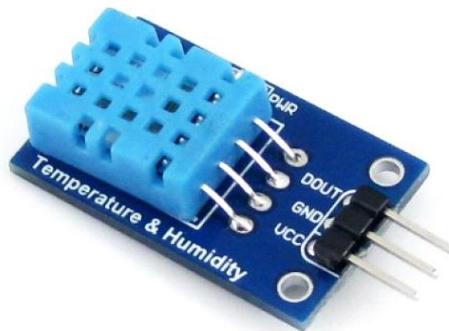


Figura 28. DHT11 (Fuente: Smoothie)

A pesar de ser un dispositivo que va conectado a un pin digital, en realidad es un sensor analógico. De hecho, es el propio sensor el que hace la conversión de analógico a digital. Una vez la señal se convierte a digital, esta se transmite hacia el microcontrolador mediante una trama de datos de 40 bits. El primer y segundo grupo de 8 bit corresponde con la parte de la humedad, correspondiendo a la parte entera y decimal, respectivamente. Lo mismo sucede con el tercer y cuarto grupo pero en este caso corresponden a la temperatura. El último grupo de 8 bit es el de paridad, que nos permite saber si hay algún error en la transmisión de la información. Los bits de paridad son el resultado de sumar los 4 primeros grupos de 8 bits, por tanto, en caso de que esta

4. Estado del arte

igualdad no se cumpla, significa que hay algún tipo de error en la transmisión. En la siguiente figura podemos ver un ejemplo (Hardware Libre, s.f.).

0011 0101	0000 0000	0001 1000	0000 0000	0100 1001
8 bits humedad	8 bits humedad	8 bits temperatura	8 bits temperatura	bits de paridad

Figura 29. Trama que contiene la información de temperatura y humedad (Fuente: Programarfacil)

- **Sensor DHT22:** El sensor DHT22 mide la temperatura y humedad al igual que su el modelo inferior, además de programarse de la misma manera. No obstante, el DHT22 mejora en diversos aspectos. En la tabla 7 se muestra una comparativa entre las especificaciones de ambos sensores.

Modelo	DHT11	DHT22
Voltaje	3,3 V a 5 V	3,3 V a 5 V
Corriente	2,5 mA	2,5 mA
Rango de temperatura	0º a 50º C ($\pm 2^{\circ}\text{C}$)	-40º a 80º C ($\pm 0,5^{\circ}\text{C}$)
Rango de humedad	20% a 80% ($\pm 5\%$ RH)	0% a 100% ($\pm 2\%$ RH)
Velocidad de lectura	1 muestra / segundo	2 muestras / segundo
Precio	8,60 €	9,99€

Tabla 7. Comparativa entre los sensores DHT11 y DHT22 (Fuente: Elaboración propia)

4. Estado del arte

Como puede verse en la tabla, la diferencia de precio es mínima y las diferencias con respecto al rango de medición, la precisión así como la velocidad de lectura son significativas. Por este motivo, para este proyecto se ha escogido el sensor DHT22.

4.4. Gateways

La red LoRaWAN así como otras tecnologías semejantes requieren de un gateway o puerta de enlace. Este dispositivo se encarga de gestionar y procesar todos los datos enviados por los nodos para, posteriormente, reenviarlos a un servidor.

Un solo gateway puede gestionar los datos transmitidos por cientos de dispositivos compatibles con LoRa al mismo tiempo. No existe una relación directa entre un nodo y un gateway, es decir, si en una zona se han desplegado varios gateways, un nodo se conectará automáticamente a uno u otro en función de distintos factores, como la distancia, la congestión de la red o la velocidad de transmisión.

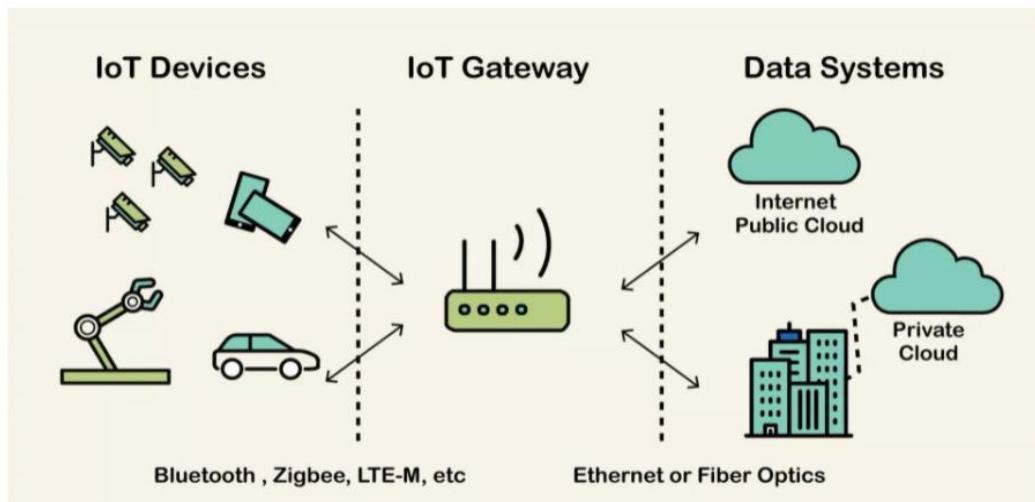


Figura 30. Esquema de una infraestructura IoT (Fuente: Lanner America)

4. Estado del arte

Los gateways pertenecientes a una red LoRaWAN siguen el siguiente proceso:

1. El gateway recibe los datos procedentes de los nodos mediante una antena que, a su vez, son demodulados mediante la electrónica de comunicaciones que lleva incorporado. Esta electrónica utiliza distintos filtros para demodular así como eliminar el ruido y otro tipo de señales que “contaminan” la información útil. Además, se emplean otros filtros que permiten eliminar el ruido y otras señales que distorsionan la información útil.
2. Una vez demodulados, los datos son encriptados en la capa de red TCP/IP SSL LoRaWAN mediante el cifrado dinámico AES-128.
3. A continuación, los datos son redirigidos hacia el servidor de TTN mediante el protocolo TCP/IP haciendo uso de la conectividad Wi-Fi, Ethernet o 3G/4G.

Las principales ventajas que aportan los gateways son:

- **Procesamiento de datos en el gateway:** El procesamiento de los datos en el gateway se asemeja al término anglosajón Edge Computing o computación en el borde. Esto permite que todos los datos se procesen antes de llegar a los servidores y de este modo solo se transmite la información realmente útil (payload), disminuyendo así la latencia y la carga de la red.
- **Incremento de la seguridad:** Un gateway es una parte fundamental de un proyecto IoT ya que no solo reduce la latencia de transmisión, sino que también añade una capa extra de seguridad. Esto es posible ya que el gateway cifra toda la información que se transmite hacia el servidor. Además, algunos gateways actúan como VPN (Virtual Private Network) (Aprendiendo Arduino, 2019), lo que permite transmitir todos los mensajes a través de un “túnel”, de modo que la posibilidad de sufrir un ataque informático disminuye de forma considerable.
- **Eficiencia energética:** Los gateways se encuentran a medio camino entre un nodo y el servidor. Esto supone un ahorro energético ya que cuanto menor sea la distancia de transmisión, menor potencia requerirá el sensor para enviar los datos (Lanner, 2019).

4.4.1. Tipos de gateways

Antes de analizar los diferentes modelos de gateways que existen en el mercado, conviene aclarar cuáles son los dos grupos en los que se dividen los gateways. Por un lado encontramos los gateways monocanal y por otro, los gateways multicanal.

- **Gateway monocanal:** Este tipo de gateway solo puede escuchar un canal y un SF (Spreading Factor) al mismo tiempo. Esto se traduce en que el rendimiento de este tipo de puertas de enlace es menor al 2% comparado con el de los gateways multicanal. Por este motivo, no están soportados oficialmente por la red The Things Network. Además, exceptuando algún modelo en concreto, no permite la comunicación downlink (Visser, 2017).
- **Gateway multicanal:** Este tipo de gateway puede escuchar de forma simultánea en 8 canales uplink y con hasta 6 diferentes SF (RAK, 2021).

A continuación, se analizarán los modelos de gateways compatibles con LoRaWAN más utilizados hoy en día.

4.4.2. RAK2245 Pi HAT + Raspberry Pi 4B

El RAK2245 Pi HAT es un módulo o concentrador que por sí solo no actúa como un gateway sino que tiene que ir conectado a una Raspberry Pi 3B o 4. A diferencia de otros concentradores similares, este modelo está diseñado para ser acoplado directamente a una Raspberry Pi sin necesidad de soldar ni añadir ningún dispositivo adicional, ya que encaja perfectamente en los 40 pines del mini PC.

Con respecto a sus especificaciones técnicas, este concentrador lleva incorporado un chip Semtech SX1301, que ofrece 8 canales para comunicación ascendente y 1 canal para comunicación descendente. Soporta hasta 500 nodos / km^2 . Además, incorpora dos chips front-

4. Estado del arte

end SX1257, que permiten la transmisión y recepción de señales tanto a alta como a baja frecuencia.



Figura 31. RAK2245 Pi HAT (Fuente: RAK)

La máxima potencia de transmisión es de 27 dBm y posee una buena sensibilidad de recepción de -139 dBm, siendo 3 dBi la ganancia de la antena LoRa. Además, lleva integrado un módulo Ublox MAX-7Q GPS para conocer en todo momento su ubicación y un disipador de calor para evitar que se sobrecaliente. Soporta las interfaces de comunicación UART, I2C y SPI y es compatible con el último protocolo de LoRaWAN, concretamente la versión 1.0.2. Las dimensiones son 56.0 x 65.0 x 22.0 mm y tiene un peso de 0.3 kg. Se alimenta mediante la Raspberry Pi a 5 V. Las frecuencias compatibles son 433 MHz, 470 MHz, 868 MHz, 915 MHz, 920 MHz y 923 MHz (RAK, 2021).

Respecto a la Raspberry Pi 4B, se trata de un pequeño ordenador que cuenta con unas características más comedidas en comparación con las de un ordenador normal aunque con un precio mucho más económico. Es el ordenador más usado en la mayoría de los proyectos de electrónica, redes o domótica a día de hoy. De hecho, se ha convertido en el ordenador más vendido en todo el mundo (Parada, 2017).

Sistema IoT de medición de CO₂ para prevención de COVID-19 basado en LoRaWAN

4. Estado del arte

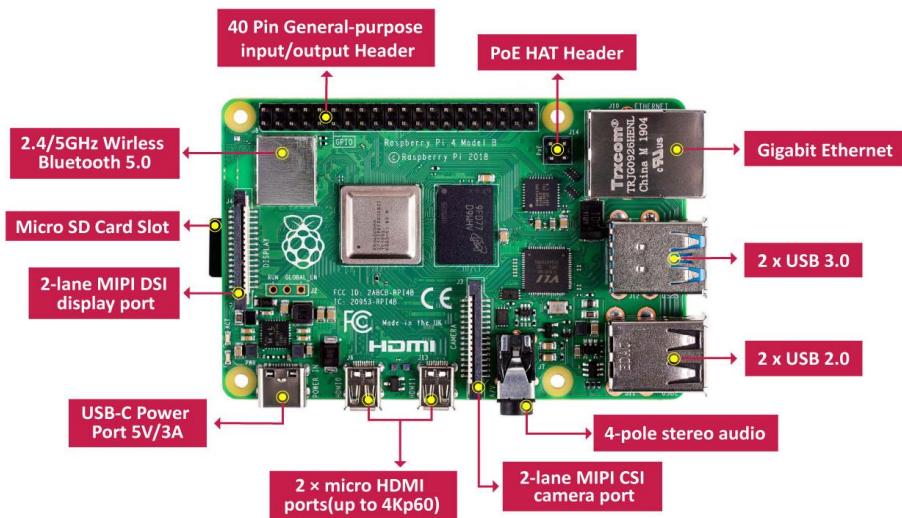


Figura 32. Componentes de la Raspberry Pi 4B (Fuente: Seeed Studio)

Especificaciones	Descripción
Procesador	Broadcom BCM2711, Quad core Cortex-A72 64-bit
Frecuencia de CPU	1,5 GHz
GPU	VideoCore VI 500 MHz
RAM	2, 4, 8 GB
Conectividad	Wi-Fi 802.11.b/g/n/ac, 2,4 GHz / 5 GHz, Bluetooth 5.0 y BLE
Puertos	Ethernet Gigabit, 40 pines, 2 x micro HDMI, 2 x USB 2.0, 2 x USB 3.0, Micro SD y USB-C
Alimentación	5 V

Tabla 8. Especificaciones de la Raspberry Pi 4B (Fuente: Raspberry Pi Foundation)

4. Estado del arte

Respecto a los sistemas son compatibles con la Raspberry Pi 4B, destacan Raspberry Pi OS (anteriormente llamado Raspbian), Ubuntu, Kali Linux, BalenaOS Windows 10 IoT o Debian (IONOS, 2020).

4.4.3. MikroTik wAP LR8 kit

A diferencia del concentrador RAK2245, el MikroTik wAP LR8 kit es un gateway que viene listo para funcionar sin necesidad de añadir un dispositivo complementario. A diferencia del RAK2245, este gateway no posee un módulo GPS. Lleva incorporado un concentrador R11e-LoRa8 MiniPCIe. Cuenta con protección IP54 lo que le permite estar ubicado en el exterior con condiciones adversas como la lluvia. 863-870 MHz es el rango de frecuencias compatible con este gateway outdoor (para exteriores). Cuenta además con una antena de 2 dBi (MikroTik, s.f.).



Figura 33. MikroTik wAP LR8 kit (Fuente: MikroTik)

4.4.4. Wirnet iStation

El Wirnet iStation es un gateway outdoor fabricado por la compañía francesa Kerlink. Cuenta con protección IP67, que se traduce en una mayor protección frente al agua. Es compatible con las frecuencias 863-874 MHz, 902-928 MHz y 915-928 MHz. Aunque no posee conectividad Wi-Fi, este gateway es compatible con Ethernet, 4G y GPS. Con respecto a la antena de LoRa, posee una ganancia de 2,6 dBi. Como vemos, posee mejores especificaciones comparado con el gateway de MikroTik (Kerlink, s.f.).



Figura 34. Wirnet iStation (Fuente: Kerlink)

4.4.5. LIG16 Indoor Gateway

El LIG16 Indoor LoRaWAN es un gateway desarrollado por Dragino y, a diferencia del resto de gateways anteriores, este es un modelo indoor, es decir, está diseñado para colocarse en interiores, por lo que no cuenta con ningún tipo de protección frente al agua. Posee un concentrador SX1302, parecido al que contiene el RAK2245 Pi HAT, que ofrece 8 canales de comunicación descendente. Las frecuencias compatibles con este gateway son 470, 863-874 y 915-928 MHz (Dragino, 2021).



Figura 35. LIG16 Indoor Gateway (Fuente: Dragino)

4. Estado del arte

4.4.6. Comparativa entre gateways LoRaWAN

La siguiente tabla muestra una comparativa entre los gateways analizados anteriormente:

Modelo	RAK2245 + RPi 4	wAP LR8 kit	Wirnet iStation	LIG16
Procesador	Broadcom BCM2711	Qualcomm Atheros QCA9531	ARM Cortex A9	Qualcomm Atheros AR9331
Frecuencia de CPU	1,5 GHz	650 MHz	2 GHz	400 MHz
Memoria Flash	32 GB (microSD)	16 MB	8 GB	16 MB
RAM	4 GB	64 MB	256 MB	64 MB
Conectividad	Wi-Fi 802.11 b/g/n/ac, 2,4 / 5 GHz, Bluetooth 5.0, BLE, Ethernet y GPS	Wi-Fi 802.11 b/g/n 2,4 GHz y Ethernet	4G, Ethernet y GPS	Wi-Fi 802.11 b/g/n 2,4GHz y Ethernet
Frecuencias soportadas	433, 470, 868, 915, 920 y 923 MHz	863-870 MHz	863-874, 902-928 y 915-928 MHz	470, 863-874 y 915-928 MHz
Sistema Operativo	BalenaOS	RouterOS	KerOS	OpenWrt
Precio	124,5 € + 64 € = 188,5 €	142,34 €	615 €	139 €

Tabla 9. Comparativa entre gateways LoRaWAN (Fuente: Elaboración propia)

4. Estado del arte

Para desarrollar este trabajo, se ha elegido la Raspberry Pi 4B junto con el concentrador RAK2245. Si bien esta opción no es la más económica, la ventaja de utilizar una Raspberry Pi en lugar de otro gateway es la capacidad de instalar otros programas adicionales como un bróker MQTT, el programa que detallaremos más adelante como es Node-RED u otro tipo de servidores.

4.5. Protocolos de comunicación IoT

Hoy en día, todavía existen multitud de sensores y dispositivos IoT que emplean una tecnología de comunicación propietaria y exclusiva, de modo que dificulta la escalabilidad así como la estandarización de aplicaciones IoT. Por este motivo, diversas empresas del sector tecnológico han creado protocolos open-source que fuesen compatibles con la mayoría de los dispositivos IoT. A continuación, se van a detallar los protocolos IoT más utilizados, así como sus características y sus principales diferencias. No obstante, antes de entrar en detalle y definir cada uno de ellos, es necesario conocer qué es un protocolo IoT así como las características que debe cumplir.

4.5.1. ¿Qué es un protocolo?

Según la RAE, un protocolo de comunicación es un “conjunto de reglas y normas que permiten que dos o más nodos de una red se comuniquen entre ellos para transmitir y recibir información” (Real Academia Española, s.f.). Para entenderlo de una forma más sencilla, un protocolo sería el “idioma” que utilizan los dispositivos IoT para “hablar” entre sí. Es una pieza fundamental ya que si no se emplearan protocolos, los dispositivos no podrían comunicarse entre sí aunque estuviesen conectados a Internet.

4.5.2. Características que debe cumplir un protocolo IoT

Algunas de las características que todo protocolo IoT posee son las siguientes (Massimi, s.f.):

4. Estado del arte

- **Escalabilidad:** Una solución IoT puede albergar decenas o incluso cientos de dispositivos y sensores, por lo que debe estar diseñada de manera que la adición de nuevos dispositivos sea un proceso rápido y sencillo. Por ello, el protocolo de comunicación debe adaptarse perfectamente a esa escalabilidad necesaria para todo tipo de proyectos.
- **Seguridad:** Cada vez son más frecuentes los ataques informáticos hacia dispositivos inteligentes y, por tanto, asegurar el protocolo de comunicación debe ser una prioridad a la hora de diseñar una infraestructura IoT. De esta manera, todos los datos que se transmitan lo harán dentro de una red cifrada.
- **Robustez:** En un sistema IoT donde concurren una gran cantidad de dispositivos, se producen multitud de conexiones al mismo tiempo. Por este motivo, el protocolo debe estar preparado para ofrecer respuestas rápidas con la menor latencia posible

A continuación se van a detallar los protocolos IoT más utilizados hoy en día en multitud de proyectos y soluciones IoT.

4.5.3. AMQP

AMQP (Advanced Message Queuing Protocol) es un protocolo open-source (de código abierto) desarrollado en 2003 por un conjunto de empresas como Huawei, Microsoft, Goldman Sachs o JPMorgan Chase, siendo esta última la que sentó las bases del protocolo. En un primer momento fue concebido para el sector financiero con el objetivo de enviar y recibir dinero de una forma más rápida y eficaz, pues en aquella época no existía ningún protocolo de código abierto que agilizara las transacciones financieras (IONOS, 2019).



Figura 36. Logo del protocolo AMQP (Fuente: Cleo)

4. Estado del arte

AMQP está basado en el protocolo TCP/IP y se desarrolla en la capa de aplicación del modelo OSI. Para encriptar y autenticar los mensajes, este protocolo utiliza el framework SASL (Simple Authentication and Security Layer, traducido como Capa de Seguridad y Autenticación Simple) así como el protocolo TLS (Transport Layer Security, traducido como Seguridad de la Capa de Transporte) (IBM, 2021).

Este protocolo destaca ofrecer un gran rendimiento por su alta fiabilidad en la transmisión de los mensajes así como una baja latencia. Además, la gestión de los mensajes se lleva a cabo mediante el encolamiento de los mismos, lo que permite almacenar los distintos mensajes hasta que el destinatario pueda recepcionarlos. Por último, la comunicación que se lleva a cabo cuando se emplea este protocolo es asíncrona, por lo que tanto el receptor como el transmisor no se comunican al mismo tiempo.

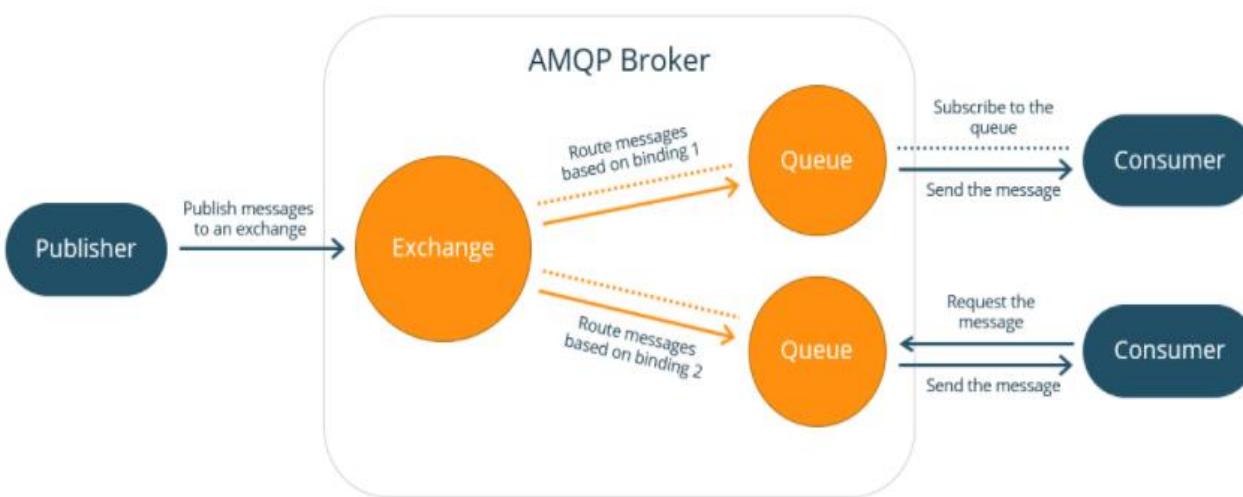


Figura 37. Arquitectura AMQP (Fuente: SmartBear)

La Figura 37 muestra las 3 partes que intervienen en el protocolo AMQP (Tezer, 2013):

- **Broker** (intermediario): Formado por el Exchange (traducido como intercambio) y las Queue (traducido como colas), el broker se encarga de gestionar el envío y recepción de los mensajes, además de encolar los mensajes.

4. Estado del arte

- **Producer o publisher** (productor): Se trata del emisor. Su función es redactar y enviar mensajes.
- **Consumer** (consumidor): Se refiere a cualquier aplicación que recibe los mensajes que estaban encolados. Cada consumer tiene asignado una cola.

Cuando el producer redacta los mensajes, este los envía posteriormente al exchange. Desde aquí, los mensajes son enrutados hasta las queue, en donde se encolan antes de ser reenviados hasta el consumer. El exchange determina la cola más adecuada para cada mensaje según unas normas llamadas bindings. El tipo de enrutamiento que lleva a cabo el broker se determina en función del tipo de exchange.

Existen 4 tipos distintos de exchanges (Tezer, 2013):

- **Direct exchange**: En este tipo de exchange, la cola de destino correspondiente a cada mensaje está definida por las routing keys o claves de enrutamiento.
- **Fanout exchange**: En este tipo de exchange, no se tiene en cuenta la routing key, sino que cada mensaje se envía a todas las colas.
- **Topic exchange**: Los mensajes son enrutados a aquellas colas cuyas routing keys coinciden con la binding routing key.
- **Headers exchange**: A diferencia del resto de exchanges, este tipo de exchange no tiene en cuenta las routing keys sino las cabeceras de los mensajes, donde se encuentran los atributos de los mismos.

Aunque AMQP se puede implementar en proyectos IoT, no está diseñado para este tipo de entornos ya que requiere de dispositivos con un gran rendimiento para garantizar la entrega de los mensajes, y los dispositivos IoT no suelen tener grandes recursos.

4. Estado del arte

4.5.4. CoAP

CoAP (Constrained Application Protocol) es un protocolo web open-source definido por la IETF (Internet Engineering Task Force) en 2014 bajo el estándar RFC 7252. Este protocolo se diseñó para posibilitar la transmisión M2M (máquina a máquina) entre dispositivos y sensores con recursos limitados y una baja capacidad de procesamiento. A diferencia del protocolo MQTT, CoAP no está basado en una comunicación donde todos los mensajes son gestionados por un nodo central (broker), sino en una comunicación descentralizada, de nodo-nodo (cliente-servidor) (Shelby, Hartke, & Bormann, 2014).

Este protocolo hace uso del modelo cliente/servidor REST, basado en HTTP. No obstante, a diferencia de este, el cual está basado en el protocolo de transporte TCP (Transmission Control Protocol), CoAP utiliza UDP (User Datagram Protocol), que permite una transmisión de mensajes 3 veces más rápida que en HTTP, ya que no está orientado a conexión y, según el tipo de mensaje enviado, no hará falta confirmar la recepción de los mismos (Arvindpdmn, 2019). La figura 38 muestra un ejemplo del funcionamiento de ambos protocolos.

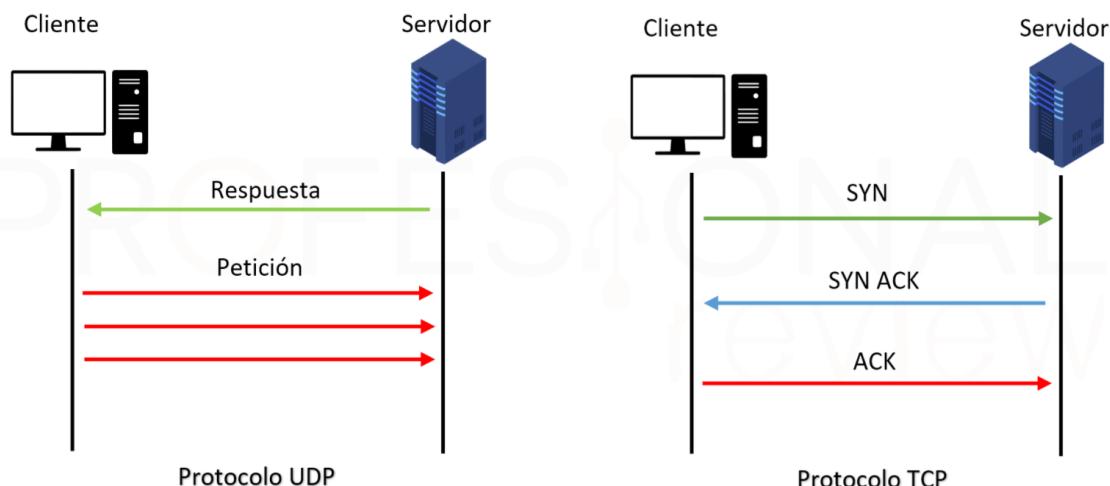


Figura 38. Comparativa entre el protocolo UDP y TCP (Fuente: Profesional Review)

4. Estado del arte

La Figura 39 muestra las arquitecturas de HTTP y CoAP, en las que intervienen los siguientes elementos:

- **Endpoint:** Hace referencia a cualquier nodo que envía o recibe un mensaje.
- **Sender:** Aquel nodo que envía un mensaje.
- **Recipient:** El nodo que recibe el mensaje.
- **Server (servidor):** El elemento que recibe una solicitud de un cliente y devuelve una respuesta.
- **Client (cliente):** En la figura aparece como C. Se trata del nodo que envía una solicitud y recibe una respuesta procedente del server.

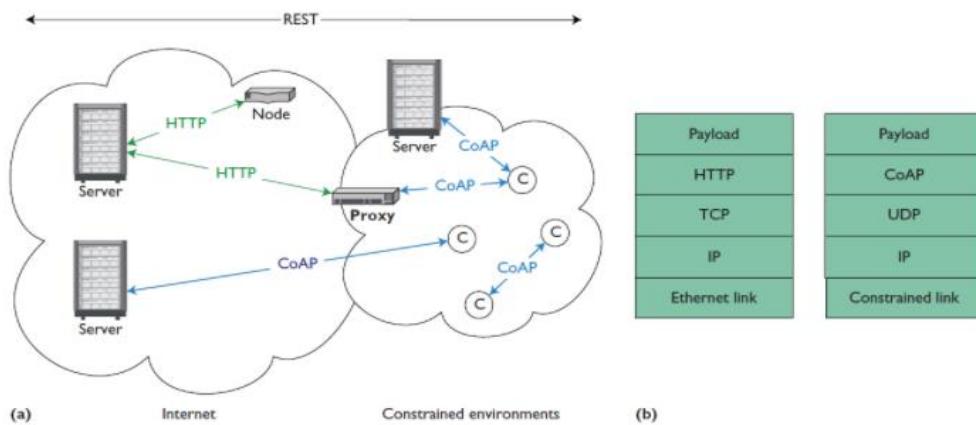


Figura 39. Arquitecturas de HTTP y CoAP (Fuente: Devopedia)

CoAP soporta 4 tipos distintos de mensajes (Azzola, 2018):

- **Confirmable message (CON)** (traducido como confirmable).
- **Non-confirmable message (NON)** (traducido como no confirmable).
- **Acknowledgment (ACK)** (traducido como reconocimiento).
- **Reset (RST)** (traducido como reinicio).

4. Estado del arte

En el caso de que se envíe un mensaje CON (Confirmable message), el emisor o sender se asegura que el mensaje enviado es recibido por el server o servidor, ya que este devuelve un mensaje confirmación ACK. En caso de que el server no haya recibido correctamente una petición, enviará un mensaje de reset o RST.

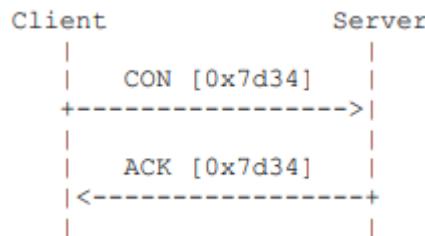


Figura 40. Transmisión confiable de mensajes (Fuente: IETF)

Si por el contrario se envía un mensaje NON (Non-confirmable), una vez que el server lo reciba, no hará falta que envíe de vuelta un mensaje ACK.

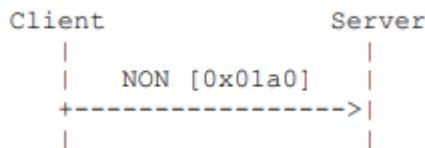


Figura 41. Transmisión no confiable de mensajes (Fuente: IETF)

Con respecto a la seguridad, CoAP emplea el protocolo DTLS (Datagram Transport Layer Security) para encriptar los mensajes transmitidos. Su funcionamiento está basado en el de TLS, aunque este solo se emplea junto con TCP. Los sistemas de cifrado compatibles con CoAP son ECC, RSA y AES (Radiocrafts, s.f.).

Las principales características del protocolo CoAP son (Radiocrafts, s.f.):

- Protocolo liviano. Permite el envío de mensajes entre dispositivos IoT con recursos limitados.
- Comunicación asíncrona y uso del protocolo de transporte UDP.

4. Estado del arte

- Basado en HTTP.
- Cabecera reducida para disminuir el tamaño del mensaje.
- Buena opción en aplicaciones que hagan uso de un servicio web.
- Protocolo idóneo para multicast y broadcast.

4.5.5. MQTT

MQTT (Message Queuing Telemetry Transport) es un protocolo de mensajería basado en publicación-suscripción (PubSub) creado por IBM en 1999 para comunicaciones máquina a máquina (M2M). En el momento de su creación se concibió como un protocolo cerrado pero en 2010 se liberó su código fuente. En 2013 se actualizó a la versión 3.1 bajo el estándar OASIS (Organization for the Advancement of Structured Information Standards), introduciendo una serie de mejoras. Actualmente es uno de los protocolos más extendidos y utilizados en todo tipo de proyectos y aplicaciones IoT (Isaac, s.f.). MQTT utiliza el protocolo TCP/IP, el cual está orientado a conexión, lo que permite una transmisión de datos más robusta y estable comparada con CoAP. La arquitectura de MQTT se basa en la topología de estrella, como se puede ver en la Figura 42.

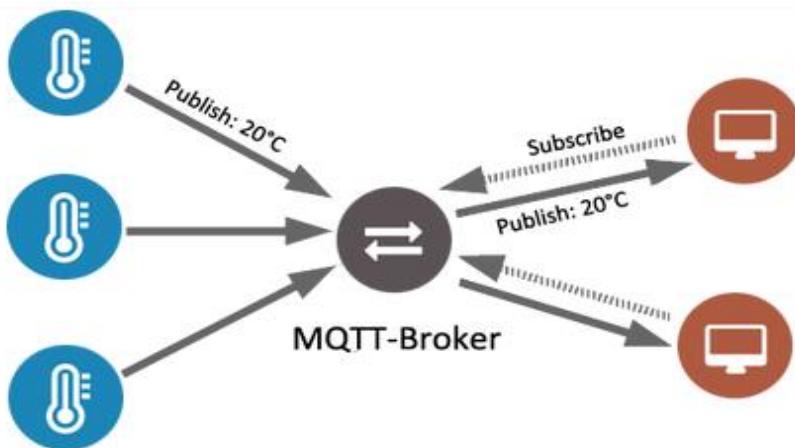


Figura 42. Arquitectura MQTT (Fuente: Eclipse)

4. Estado del arte

En una comunicación donde se utiliza el protocolo MQTT, intervienen los siguientes elementos (The HiveMQ Team, 2019):

- **Client** (traducido como cliente): Son los nodos que se conectan a un broker para recibir o enviar mensajes.
- **Broker**: Se trata de un servidor que gestiona todos los mensajes que envían los clientes.
- **Tópico**: Hace referencia al tema del mensaje. Es decir, se trata de un filtro que permite diferenciar los mensajes que se envían en función de la información que estos contienen.
- **Payload** (traducido como carga útil): Es el contenido del mensaje.

➤ Funcionamiento del protocolo MQTT

En primer lugar, el cliente emisor establece una conexión TCP/IP con el broker mediante la URL o dirección IP asociada al servidor junto con el puerto en el que esté disponible. Normalmente el puerto de comunicación empleado es el 1883 para conexiones sin cifrado y el 8883 para conexiones que utilicen el protocolo de seguridad SSL/TLS. El primer mensaje que envía el cliente se denomina CONNECT, y contiene información como el ID, nombre de usuario y contraseña del cliente. El broker aceptará o rechazará la conexión con el cliente mediante un mensaje llamado CONNACK.

En segundo lugar, si se ha establecido la conexión entre ambos, el cliente emisor utilizará los paquetes PUBLISH para transmitir la información hacia el broker. Este tipo de paquetes incluye el tópico y el payload de los mensajes.

4. Estado del arte

Los tópicos se estructuran en base a una jerarquía o niveles separados por una barra “/”:

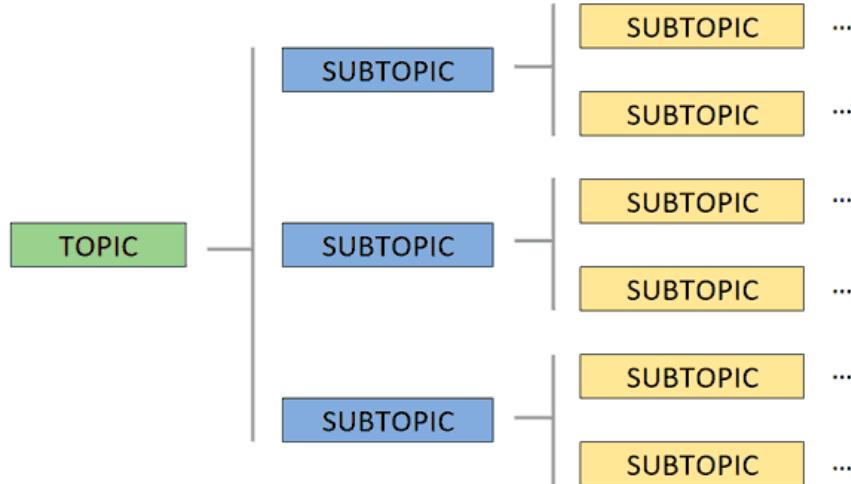


Figura 43. Diferentes niveles de tópicos (Llamas, 2019)

Un ejemplo de una jerarquía de tópicos sería:

Hotel/Habitación/Armario

Hotel/Cocina/Frigorífico

Hotel/Cocina/Horno

Cuando uno o varios clientes desean recibir mensajes de un tópico en concreto, deberán hacer uso de los paquetes SUBSCRIBE indicando el tópico deseado. En caso de no querer suscribirse únicamente a un tópico sino a varios, el cliente debe usar wildcards, una serie de reglas que se utilizan en MQTT para una mejor gestión de los tópicos (The HiveMQ Team, 2019):

- **Single level wildcard** (nivel único): Se representa con el carácter "+" y se utiliza para recibir un subtópico de un nivel determinado que se repite en distintos tópicos. Por ejemplo, tenemos la siguiente jerarquía:

Garaje/Coché/Color

4. Estado del arte

Garaje/**Moto**/Color

Coche/**Patinete**/Color

Si un cliente desea recibir todos los tópicos Color, tendrá que escribir:

Garaje/+/Color

Si en cambio desea recibir todos los tópicos de un nivel en concreto:

Garaje/Coche/**Color**

Garaje/Coche/**Marca**

Debe escribir: **Garaje/Coche/+**

- **Multi level wildcard** (nivel múltiple): Se utiliza con el carácter # y permite suscribirse a todos los subtópicos a partir de un nivel determinado. Por ejemplo, tenemos los siguientes tópicos:

Edificio/Planta/**Oficina/Despacho**

Edificio/Planta/**Oficina/Aseo**

Edificio/Planta/**Tienda/Probador**

En caso de que un cliente desee recibir todos los payload cuyo tópico comience con Edificio/Planta, tendrá que escribir lo siguiente:

Edificio/Planta/#

Si por el contrario un cliente desea recibir todos los mensajes que se envían a un broker MQTT, únicamente deberá suscribirse al tópico #.

4. Estado del arte

Para dejar de recibir mensajes procedentes de un determinado tópico, se utiliza el paquete UNSUBSCRIBE. El broker utilizará los paquetes SUBACK y UNSUBACK para indicar al cliente si se ha podido suscribir o no al tópico deseado. En tercer lugar, para comprobar si la conexión sigue abierta o se ha cerrado, los clientes envían regularmente un mensaje PINGREQ. El broker responderá a este mensaje mediante un paquete PINGRESP. Por último, cuando un cliente quiera desconectarse del broker, enviará un mensaje DISCONNECT (The HiveMQ Team, 2019).

La Figura 44 muestra de gráficamente un resumen de todo el proceso descrito anteriormente.

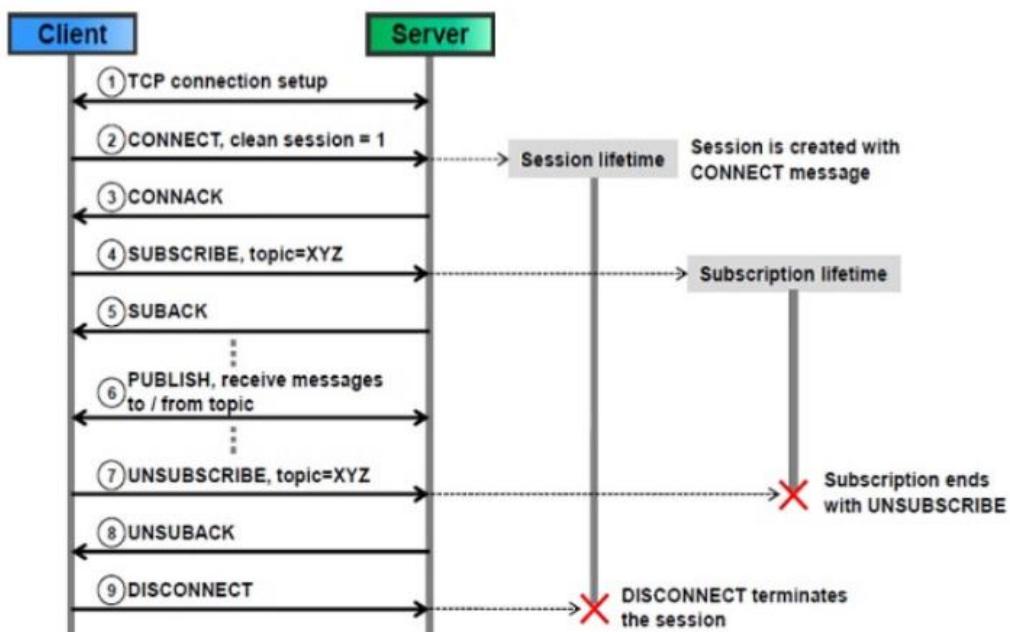


Figura 44. Proceso para recibir mensajes de un tópico concreto (Fuente: Devopedia)

➤ Calidad del servicio (QoS)

A diferencia de otros protocolos IoT, MQTT destaca por la calidad de servicio (QoS). Esta característica permite al usuario elegir entre 3 niveles distintos de servicio en función de la latencia o la robustez deseada en la transmisión de mensajes. A continuación, se describen las características de cada uno de ellos (IBM, 2021):

4. Estado del arte

- **QoS 0** (Fire and forget o Disparar y olvidar): Se trata del nivel más sencillo y ligero pues el mensaje sólo se envía una única vez. En caso de que hubiese algún tipo de error, el mensaje se pierde. El receptor no confirma al emisor que ha recibido el mensaje. Este tipo de nivel es interesante usarlo cuando la información que queremos enviar no es crítica.
- **QoS 1** (Al menos una vez): En este nivel el mensaje se envía como mínimo 1 vez. El receptor confirma la recepción del mensaje PUBLISH mediante el paquete PUBACK. Este tipo de nivel está indicado para situaciones en las recibir duplicado un mensaje no es un problema.
- **QoS 2** (Exactamente una vez): Este nivel garantiza que el mensaje será entregado una única vez. Esto es posible gracias al uso de 2 pares: PUBLISH/PUBREC y PUBREL/PUBCOMP. Este nivel se debe emplear en situaciones críticas donde no es posible recibir mensajes duplicados ni pérdidas de los mismos.

Todos los mensajes enviados que hagan uso del nivel de calidad QoS 1 o QoS 2 son encolados hasta que un cliente los recepcione siempre y cuando tenga habilitada una sesión persistente. La Figura 45 muestra de forma esquemática los paquetes que intervienen en cada uno de los distintos niveles:

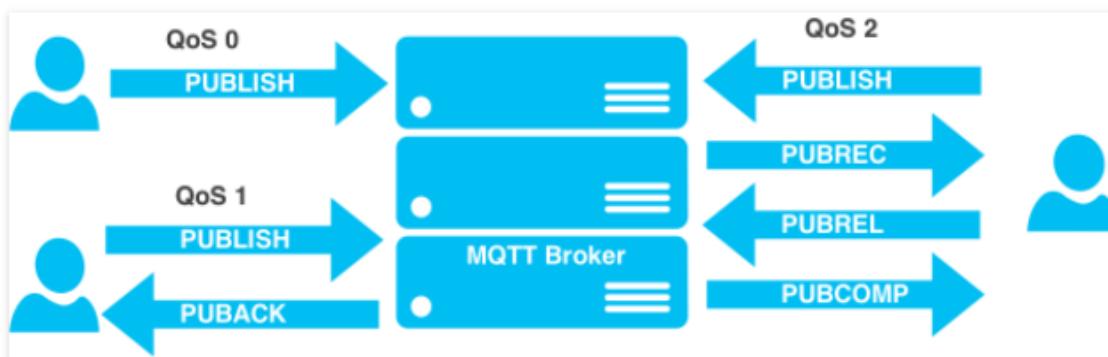


Figura 45. Niveles de calidad de servicio (QoS) (Fuente: Aprendiendo Arduino)

4. Estado del arte

Las principales características del protocolo MQTT son (Isaac, s.f.):

- Ligero. Se trata de un protocolo diseñado para dispositivos con recursos limitados y una transmisión de mensajes con un bajo ancho de banda.
- Rápido, eficiente y bajo consumo energético.
- Escalable, ya que permite añadir nuevos dispositivos al broker de un modo muy sencillo.
- Transmisión segura de la información gracias a los protocolos SSL/TLS.
- Comunicación asíncrona.

4.5.6. Comparativa entre protocolos IoT

Una vez se han descrito los 3 protocolos IoT más utilizados en la actualidad, se van a comparar las principales características de cada uno en la siguiente tabla:

Protocolo	AMQP	CoAP	MQTT
Transporte	TCP/IP	UDP/IP	TCP/IP
Modelo	Intercambio de mensajes punto a punto	Petición-Respuesta (REST)	Publicación-Suscripción
Seguridad	SASL/TLS	DTLS	SSL/TLS
Tolerancia a fallos	Específica de la implementación	Descentralizado	El broker es el único punto de fallo (SPoF)
Enfoque	Aplicaciones robustas, fiables y con baja latencia	Dispositivos con recursos limitados y aplicaciones REST	Muchos dispositivos interconectados con baja capacidad de procesamiento

Tabla 10. Comparativa entre protocolos IoT (Fuente: Startup Training)

4. Estado del arte

Para este proyecto se ha elegido el protocolo MQTT ya que es uno de los más ligeros y rápidos que, además requieren de poca potencia para funcionar. Asimismo, existen multitud de aplicaciones compatibles con este protocolo, como por ejemplo Node-RED (la cual se explicará en el apartado 4.7) así como el IDE de Arduino. Además, la instalación de un broker MQTT y su integración con dispositivos como Arduino o ESP32 es más sencilla en comparación con AMQP o CoAP.

4.6. The Things Network

La tecnología LoRa por sí misma permite únicamente la transmisión de datos de punto a punto. En determinadas ocasiones esto puede ser útil, aunque es insuficiente para crear una red IoT ya que los datos no se envían a Internet. Para lograrlo, se recurre a la tecnología LoRaWAN, en la que cualquier sensor compatible con LoRa puede transmitir y recibir datos que provengan de Internet mediante un gateway. La incorporación de nuevos gateways y dispositivos a LoRaWAN así como los datos transmitidos por estos, son gestionados por The Things Network (TTN).

The Things Network es una red IoT pública que actúa como un servidor y permite la interconexión de dispositivos LoRa a Internet mediante puertas de enlace o gateways. Creada en Ámsterdam en 2015 por Johan Stokking y Wienke Giezeman, esta red se diseñó como una alternativa gratuita y descentralizada a las redes IoT que había en ese momento, las cuales eran propietarias y había que pagar por su uso (Medina, 2019). Existen otras redes privadas open-source como ChirpStack que, a diferencia de TTN, permiten un control absoluto de la red, sin depender de un servidor público (Alfa IOT, 2021). No obstante, para este TFG, se ha elegido la red TTN por su facilidad y su compatibilidad con aplicaciones de terceros.

Como se puede ver en la Figura 46, dentro de la arquitectura LoRaWAN, la red TTN conecta el gateway y las plataformas IoT para analizar y visualizar los datos. En la figura, TTN aparece como *Network Server* y *Application Server*.

4. Estado del arte

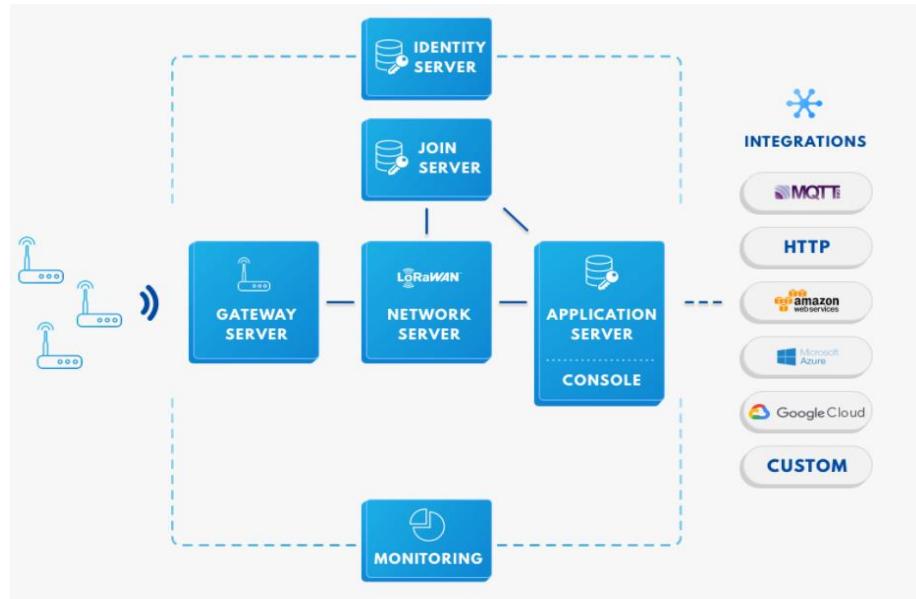


Figura 46. Arquitectura LoRaWAN (Fuente: TTN)

De esta forma, los pilares en los que se fundamenta la red TTN son (Victor, 2019):

- **Red pública y gratuita:** Cualquier nodo situado a una distancia inferior a 15 km en zonas rurales o 5 km en zonas urbanas con respecto a un gateway LoRaWAN puede transmitir datos a Internet de forma gratuita mediante la tecnología LoRa.
- **Red escalable:** Cualquier persona puede formar parte de la comunidad de The Things Network y proporcionar cobertura LoRa en aquellas zonas donde no esté disponible, favoreciendo así el uso de esta red IoT. Actualmente, existen más de 20600 gateways repartidos en 151 países (The Things Network). La figura 43 muestra cómo están repartidos los gateways en todo el mundo.
- **Visualización de datos:** Además de gestionar los datos recibidos por los sensores, TTN ofrece una interfaz para poder visualizarlos.
- **Compatibilidad con diferentes plataformas y protocolos:** TTN se creó para que fuese totalmente compatible con otras plataformas IoT como por ejemplo Ubidots o Thingspeak mediante integraciones directas así como con los protocolos MQTT o HTTP. Además,

4. Estado del arte

The Things Network ofrece APIs que pueden ser utilizadas en distintos lenguajes de programación como por ejemplo Python, Node.js, Go o Java.

- **Seguridad:** A pesar de ser una red IoT pública, The Things Network se diseñó para ofrecer una alta seguridad ya que cifra de extremo a extremo los datos transmitidos por los nodos. Además, soporta el cifrado AES (Advanced Encryption Standard) de 128 bits.



Figura 47. Distribución de gateways LoRaWAN por todo el mundo (Fuente: TTN)

4.7. Plataformas IoT

Todos los apartados analizados anteriormente como las placas de desarrollo, las tecnologías o los protocolos de comunicación IoT son claves cuando se trata del diseño y la implementación de una red IoT. No obstante, entre todos ellos, la parte más importante son los datos. En efecto, la obtención de información procedente de sensores y, su posterior análisis y procesado, es el objetivo final de toda solución IoT. Para facilitar esta tarea, multitud de compañías tecnológicas relacionadas con el sector del IoT han desarrollado distintas plataformas y herramientas que permiten visualizar y almacenar datos. Algunos son open-source y ofrecen versiones gratuitas aunque sacrificando algunas características, mientras que otros son programas con licencia y suelen tener un coste mensual.

En este punto, se analizarán en detalle las plataformas IoT que mejor se integran con The Things Network, como son Datacake, Ubidots, ThingSpeak y Node-RED.

4.7.1. Datacake

Datacake es un software basado en la gestión de dispositivos IoT y la visualización de diferentes datos, diseñado para ser utilizado sin conocimientos previos de programación. Al igual que el resto de las plataformas IoT, ofrece diferentes herramientas como gráficas, widgets o medidores para poder visualizar toda la información procedente de los sensores. Una característica destacable de esta plataforma es que a diferencia de otros dashboard IoT, Datacake integra un broker MQTT de forma nativa, lo cual favorece positivamente a su escalabilidad ya que existe la posibilidad de recibir datos procedentes de cualquier broker MQTT y, a su vez, reenviarlos a otro distinto (<https://docs.datacake.de/>).



Figura 48. Logo de Datacake (Fuente: Datacake)

4. Estado del arte

La gestión de los datos y dispositivos se lleva a cabo mediante un elemento llamado “Product” (traducido como Producto), el cual contiene los diferentes campos de los dispositivos, una base de datos donde se almacenan los datos así como otros datos relativos al método de conexión, ya sea mediante integración directa o por MQTT. Un Producto puede albergar varios dispositivos facilitando así su gestión ya que todos comparten los mismos campos y otras configuraciones. Es decir, cada vez que se añade un nuevo dispositivo a un Producto ya existente, este hereda todos los campos ya creados previamente. La estructura de un Producto se resume de la manera:

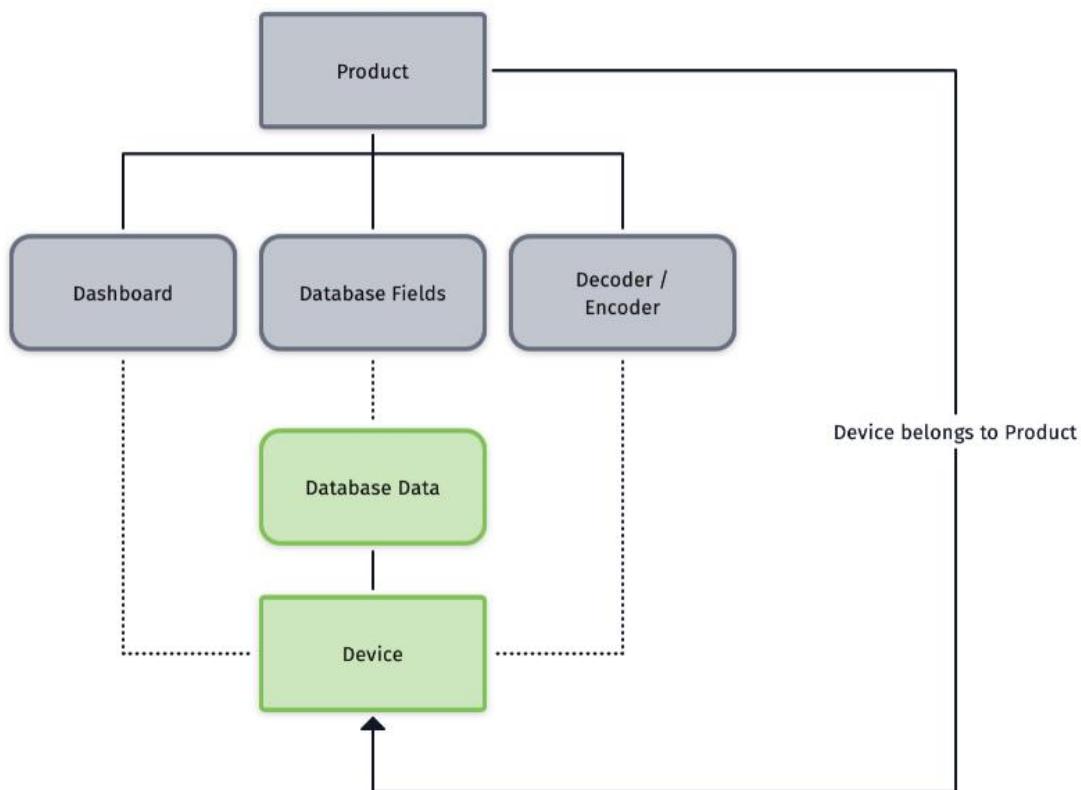


Figura 49. Estructura de un Product de Datacake (Fuente: Datacake)

4. Estado del arte

El resto de las características que ofrece esta plataforma IoT son (<https://docs.datacake.de/>):

- Exportar los datos almacenados en formato Excel, CSV o JSON.
- Generar enlaces para que otros usuarios puedan visualizar los dashboard de forma pública. Esto cobra especial relevancia cuando queremos compartir información de forma pública y transparente sin necesidad de registrarse en la plataforma.
- Añadir colaboradores a un dashboard para que puedan añadir o modificar diferentes parámetros.
- Posibilidad de utilizar la plataforma Cake Red basada en Node-RED pagando una tarifa mensual.
- Posibilidad de crear alertas mediante SMS o correo electrónico cuando por ejemplo se supere un cierto valor.

En función del número de dispositivos que se deseen añadir a la plataforma IoT, Datacake pone a disposición de los usuarios diferentes planes los cuales aparecen en la siguiente tabla:

Planes	Free	Light	Standard	Plus
Precio / mes	Gratis	1€	3€	5€
Dispositivos	2	1	1	1
Datapoints / día	500	1.000	2.500	7.500

Tabla 11. Tabla comparativa de los distintos planes ofertados por Datacake (Fuente: Datacake)

4. Estado del arte

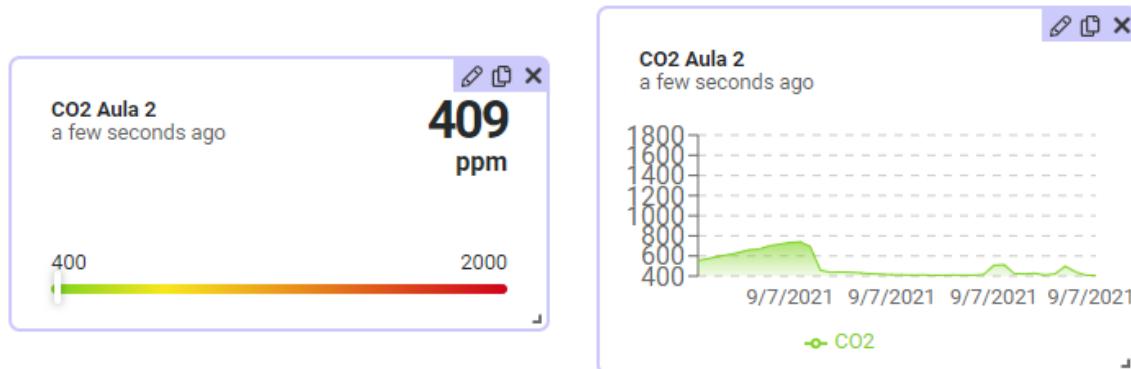


Figura 50. Widget que ofrece el dashboard Datacake (Fuente: Elaboración propia)

4.7.2. Ubidots

Fundada en 2012 por Agustín Pelaez, Ubidots es una plataforma IoT que permite almacenar y mostrar de manera gráfica los datos procedentes de sensores IoT (<https://ubidots.com/about/>). Ofrece una interfaz sencilla y no se requieren conocimientos previos de programación para su uso, lo que permite que cualquier usuario pueda utilizarlo sin problemas.



Figura 51. Logo de Ubidots (Fuente: Ubidots)

Ubidots es compatible con los protocolos MQTT y HTTP, y permite recibir datos procedentes de dispositivos LoRa mediante una integración directa con la red The Things Network. Además, dispone de una serie de APIs que facilitan su integración con otras aplicaciones compatibles con los lenguajes de programación C#, Javascript, Ruby y Python. Ofrece también la opción de personalizar notificaciones y alertas vía email, Telegram, SMS o incluso mediante llamada telefónica (<https://ubidots.com/about/>).

4. Estado del arte

Ubidots utiliza los *dots* (traducido como puntos) para medir la cantidad de datos que envían los dispositivos. Cada vez que se actualiza una variable de un sensor, esto cuenta como un nuevo dot. Por ejemplo, si un dispositivo cuenta con un sensor que envía la temperatura medida a Ubidots cada 30 minutos, este sensor habrá generado cada mes:

$$1 \text{ dot} \times 2 \text{ veces / hora} \times 24 \text{ horas} \times 30 \text{ días} = 1440 \text{ dots / mes}$$

En función del plan elegido, el límite máximo de dots generados variará. Con respecto a los dispositivos que se pueden registrar en Ubidots, existen más de 30 marcas cuyos dispositivos son compatibles, entre las cuales destacan las siguientes:

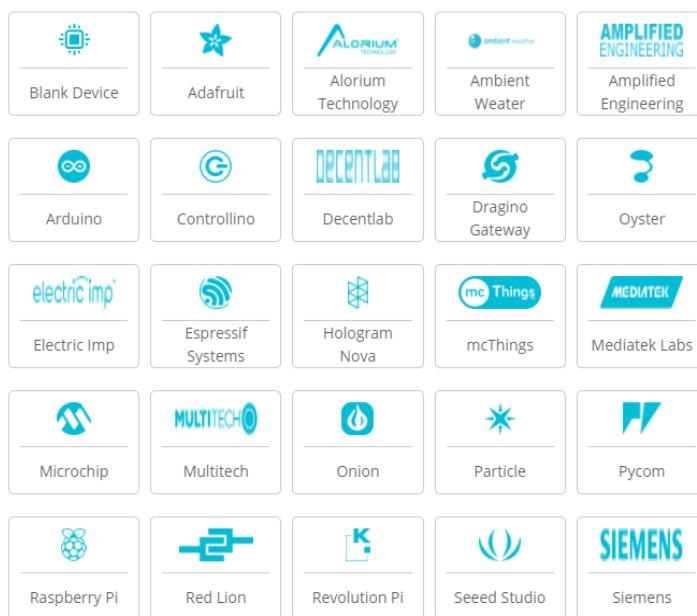


Figura 52. Marcas compatibles con Ubidots (Fuente: Ubidots)

Una vez escogida la marca de nuestro dispositivo, Ubidots ofrece distintas herramientas y widgets para poder visualizar y graficar los datos recibidos. La Figura 53 muestra todas las opciones disponibles:

Sistema IoT de medición de CO₂ para prevención de COVID-19 basado en LoRaWAN

4. Estado del arte

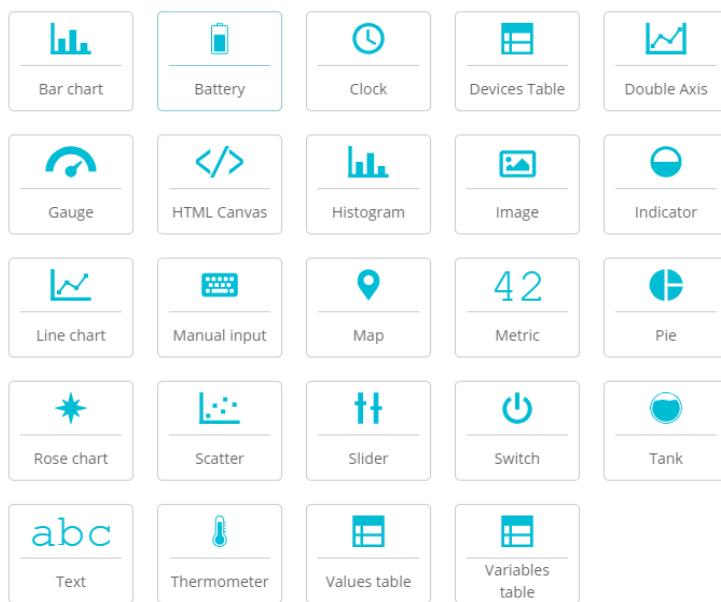


Figura 53. Widgets que ofrece Ubidots (Fuente: Ubidots)

Con respecto al apartado económico, Ubidots oferta los siguientes planes:

Planes	STEM	IoT Entrepreneur	Professional	Industrial	Scale
Precio	Gratis	\$49	\$199	\$499	\$1.799
Dispositivos	3	25	200	1.000	4.000
Dots máximos / mes	5.000	2.000.000	15.000.000	50.000.000	200.000.000
Usuarios finales	-	-	50	200	800
Dispositivos adicionales	-	-	\$1 / dispositivo	\$0,50 / dispositivo	\$0,40 / dispositivo

Tabla 12. Comparativa de los distintos planes ofrecidos por Ubidots (Fuente: Ubidots)

4.7.3. ThingSpeak

ThingSpeak es una plataforma IoT que pertenece a MathWorks, compañía estadounidense propietaria del software MATLAB. Esta aplicación permite recoger y almacenar información procedente de dispositivos y sensores en tiempo real. Además, el hecho de pertenecer a la misma empresa que MATLAB, ofrece la posibilidad de analizar y procesar los datos mediante algoritmos de Machine Learning (ThingSpeak, s.f.).

Toda la información llega a la plataforma mediante los protocolos MQTT o HTTP, o bien a través de su API. Tal y como sucede en Ubidots, en ThingSpeak también existe la posibilidad de recibir los datos mediante una integración directa de The Things Network.

Una característica única que ofrece es la posibilidad de crear plugins personalizados por los usuarios. Los plugins permiten desarrollar aplicaciones dentro de la propia plataforma en distintos lenguajes como Javascript o HTML y de este modo, añadir mejoras a los paneles donde se visualizan los datos. Respecto a los dispositivos soportados, destacan Raspberry Pi, Arduino, BeagleBone Black y ESP8266 (Descubre Arduino, s.f.).

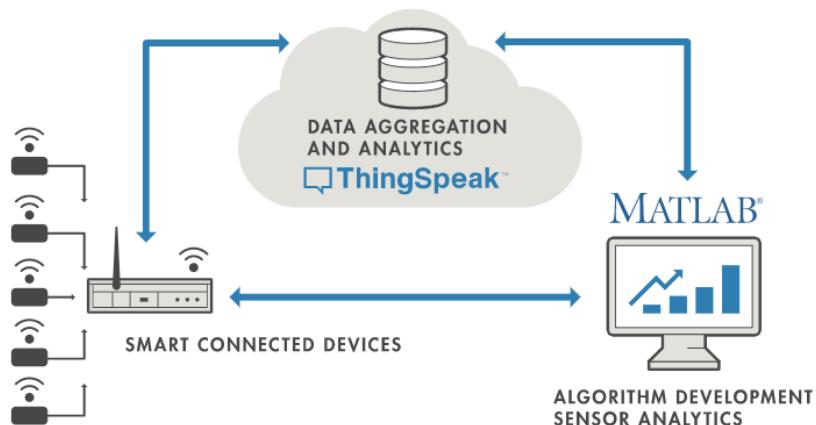


Figura 54. Arquitectura de ThingSpeak (Fuente: ThingSpeak)

4. Estado del arte

El funcionamiento de ThingSpeak se fundamenta en un elemento llamado “Canal ThingSpeak”, el cual permite almacenar los datos transmitidos por los sensores. Algunos parámetros que forman un canal son (Descubre Arduino, s.f.):

- Nombre del canal.
- Descripción.
- 8 campos para almacenar cualquier tipo de dato, procedentes de un sensor.
- 3 campos para almacenar la elevación, longitud y latitud.
- 1 campo que contiene la información sobre el tipo de datos almacenados.

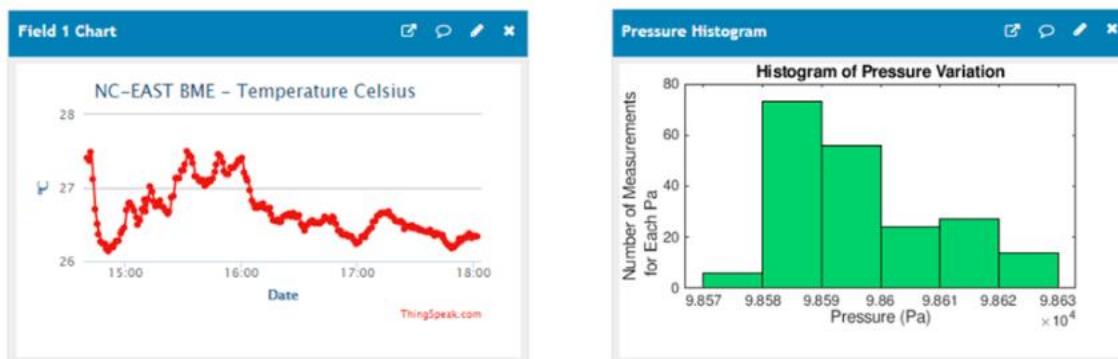


Figura 55. Ejemplo de un dashboard creado mediante ThingSpeak (Fuente: ThingSpeak)

4. Estado del arte

New Channel

Name	<input type="text"/>	Elevation	<input type="text"/>
Description	<input type="text"/>	Show Channel Location	<input type="checkbox"/>
Field 1	<input type="text"/> <input type="checkbox"/>	Latitude	<input type="text"/> 0.0
Field 2	<input type="text"/> <input type="checkbox"/>	Longitude	<input type="text"/> 0.0
Field 3	<input type="text"/> <input type="checkbox"/>	Show Video	<input type="checkbox"/>
Field 4	<input type="text"/> <input type="checkbox"/>	<input checked="" type="radio"/> YouTube <input type="radio"/> Vimeo	
Field 5	<input type="text"/> <input type="checkbox"/>	Video URL	<input type="text"/> http://
Field 6	<input type="text"/> <input type="checkbox"/>	Show Status	<input type="checkbox"/>
Field 7	<input type="text"/> <input type="checkbox"/>	Save Channel	
Field 8	<input type="text"/> <input type="checkbox"/>		
Metadata	<input type="text"/>		
Tags	<input type="text"/> (Tags are comma separated)		
Link to External Site	<input type="text"/> http://		
Link to GitHub	<input type="text"/> https://github.com/		

Figura 56. Partes que componen un canal de ThingSpeak (Fuente: ThingSpeak)

4. Estado del arte

Al igual que en Ubidots, ThingSpeak ofrece distintos planes para hacer uso de su software en función de las características que nos interesen. Los planes ofertados son:

Planes	FREE	STUDENT	HOME	ACADEMIC	STANDARD
Tipo de proyecto	No comercial	Estudiantes	Uso personal	Profesores y personal académico	Uso comercial
Nº mensajes / año	3.000.000	33.000.000	33.000.000	33.000.000	33.000.000
Nº canales	4	10	10	250	250
Suscripciones máx. de MQTT	3	50	50	50	50
Precio / año	Gratis	55 €	75 €	250 €	600 €

Tabla 13. Tabla comparativa de los distintos planes ofertados por ThingSpeak (Fuente: ThingSpeak)

4.7.4. Node-RED

Node-RED es un entorno open-source de programación visual que permite interconectar dispositivos IoT de distintas marcas así como diferentes protocolos de comunicación. Creado por los empleados de IBM, Dave Conway Jones y Nick O'Leary en el 2013, este software basado en Node-JS se diseñó para ser muy sencillo, intuitivo y fácil de utilizar. Cualquier usuario puede utilizar este programa pues no se requieren grandes nociones de programación, aunque si queremos hacer aplicaciones más complejas, debemos tener conocimientos básicos de JavaScript (del Valle Hernández, s.f.).

Su interfaz está basada en nodos que a su vez se agrupan en flows (flujos), pequeños programas que realizan una o varias tareas. Existe una gran cantidad de servicios y dispositivos compatibles con Node-RED entre los cuales se encuentran Alexa, Google Home, Raspberry Pi,

4. Estado del arte

ThingSpeak, Telegram, ThingerIO, IFTTT, etc. Además, los protocolos MQTT y HTTP, así como TCP y UDP también son compatibles. Cada vez son más las empresas que desarrollan sus propios nodos dentro de Node-RED para poder escalar y maximizar sus opciones de desarrollo (del Valle Hernández).

Cabe resaltar que todas las herramientas que ofrece este software se pueden usar gratuitamente, a diferencia de Ubidots y Thingspeak.

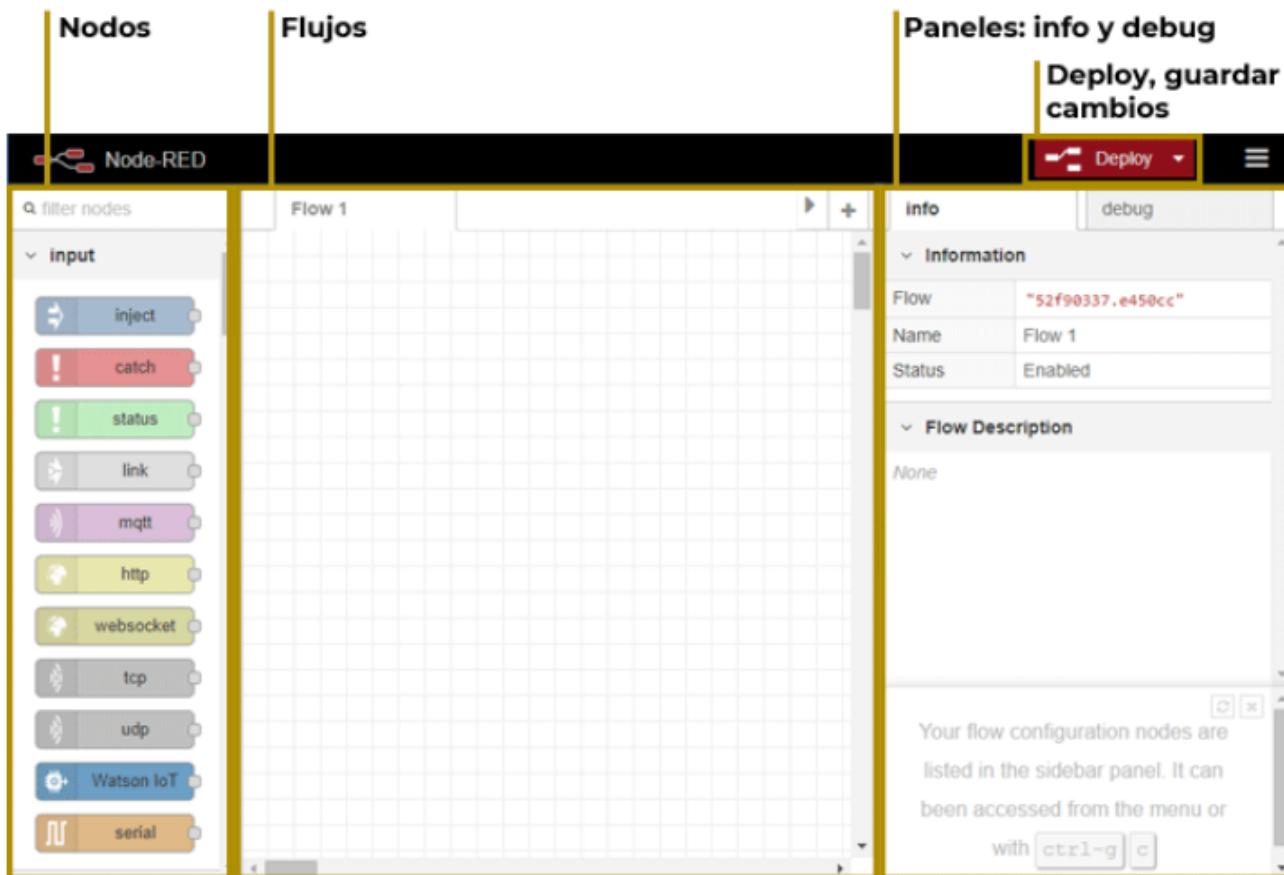


Figura 57. Interfaz gráfica de Node-RED (Fuente: Programar Fácil)

4. Estado del arte

Como se puede apreciar en la Figura 58, la interfaz gráfica se divide principalmente en 3 partes:

- **Nodos:** En este apartado se encuentran todos los nodos que podemos utilizar, los cuales están separados por categorías, como muestra la Figura 59.
- **Flujos:** En esta zona podemos añadir nodos y crear flujos para realizar determinadas tareas. La Figura 60 muestra un flujo compuesto por un nodo inject (traducido como inyectar) y debug (traducido como depurar). Debemos conectarlos mediante una línea para que puedan interactuar uno con otro. Cada vez que apretemos el botón del nodo inject, aparecerá en el panel debug un payload o mensaje que hayamos determinado previamente
- **Paneles:** En esta sección podemos ver diferentes pestañas:
 - **Help:** Muestra la definición de cada uno de los nodos que estamos utilizando.
 - **Debug:** Muestra todos los registros y datos generados por los nodos. Es decir, si estamos recibiendo información procedente de un broker MQTT, en esta zona podemos ver todos los mensajes recibidos.
 - **Info:** Muestra una lista de todos los flujos y nodos que estamos utilizando.
 - **Config:** Desde esta pestaña se accede a la configuración de los nodos relacionados con los servidores y bases de datos.
 - **Dashboard:** Se pueden modificar los parámetros del dashboard que ofrece Node-RED de forma nativa.

4. Estado del arte

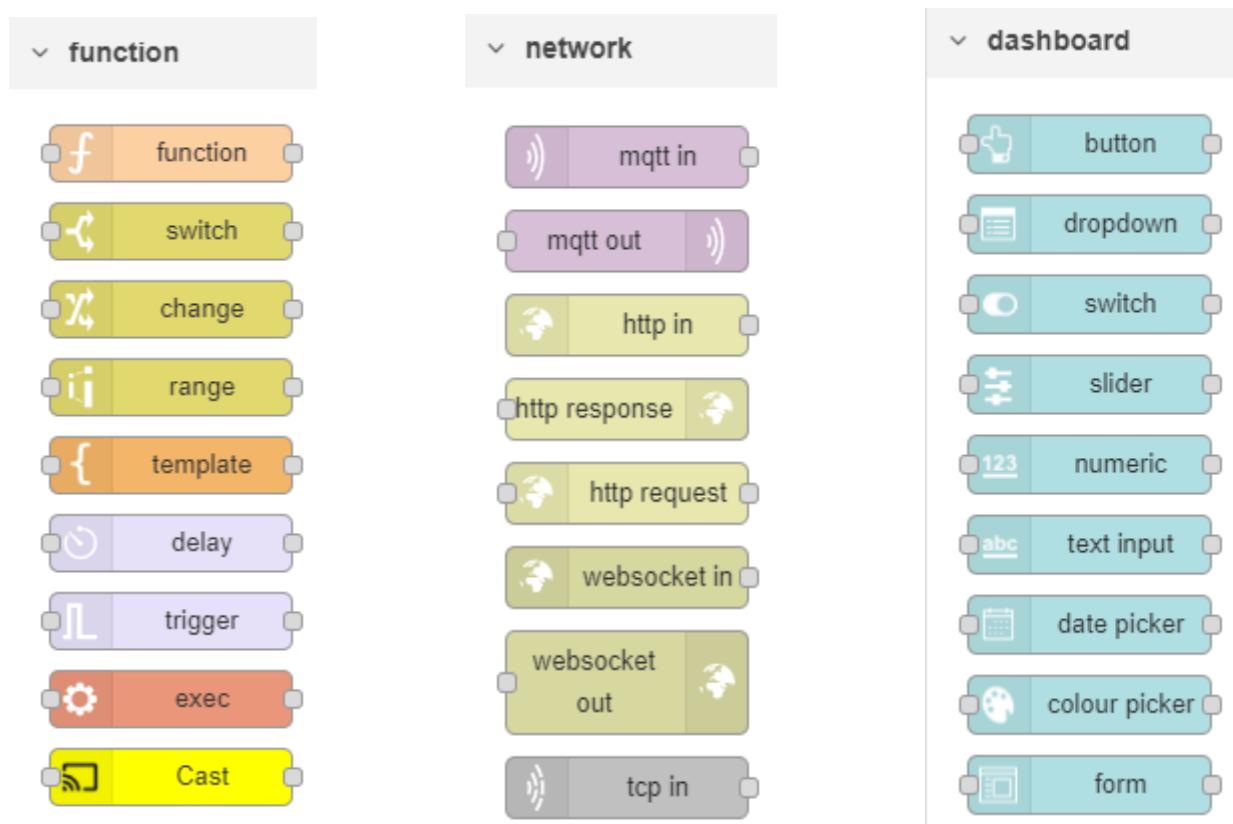


Figura 58. Node-RED (Fuente: Elaboración propia)



Figura 59. Flujo de Node-RED (Fuente: Elaboración propia)

4. Estado del arte

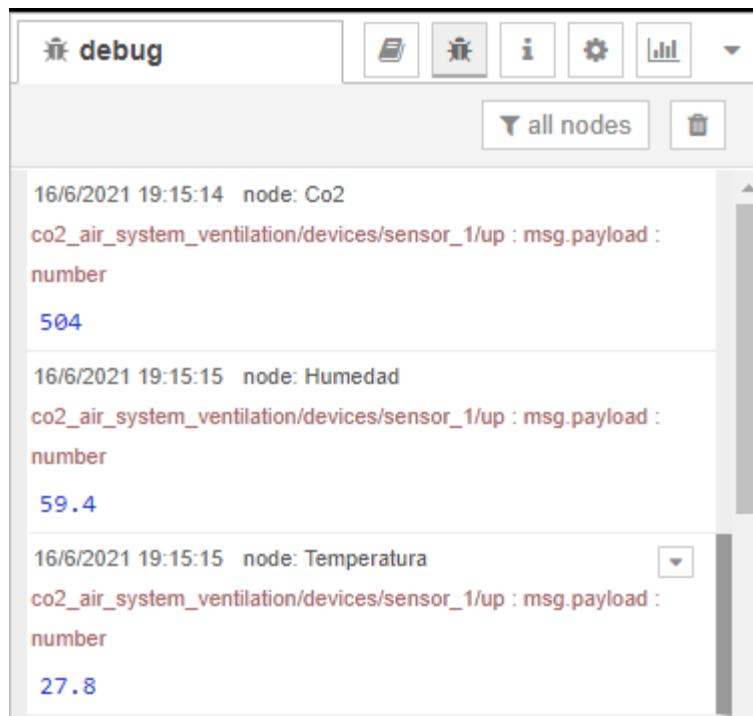


Figura 60. Pestaña de debug en Node-RED (Fuente: Elaboración propia)

Node-RED se utilizará para gestionar los mensajes MQTT procedentes del broker de TTN, así como las alertas junto con la aplicación de mensajería Telegram. No se ha analizado ningún otro programa similar ya que, a día de hoy, no existe otro software que ofrezca de forma gratuita estas funcionalidades y herramientas.

5. Desarrollo de la solución

Una vez se han analizado todos los apartados correspondientes al Estado del Arte, se va a detallar la solución propuesta, la cual se divide en 4 diferentes partes:

- Nodos
- Gateway
- Servidor TTN
- Plataforma IoT

5.1. Especificaciones y estructura de la solución

En este apartado se van a detallar los requisitos necesarios para cumplir los objetivos mínimos que se plantean en este TFG.

- **Nodo:** El nodo o dispositivo final debe contar con una placa de desarrollo compatible con LoRa, una pantalla para poder visualizar los diferentes parámetros, los sensores de temperatura, humedad y CO₂, y una antena de 868 MHz para poder transmitir la información hacia el gateway. Con el objetivo de proteger la todos los componentes, se propone construir una carcasa mediante impresión 3D dado que de este modo, las piezas pueden diseñarse a medida.
- **Gateway:** El gateway LoRaWAN debe ser multicanal para que pueda escuchar en 8 canales simultáneamente, y así ofrecer cobertura LoRa a todos los nodos que estén en su rango de cobertura aunque la red esté congestionada. Para este TFG, es suficiente con un gateway indoor ya que se instalará en una zona bajo techo y, además, supone un menor coste comparado con aquellos que pueden instalarse en exteriores.
- **Servidor TTN:** Es el servidor que se ha elegido para gestionar los datos transmitidos por los nodos. Es open-source y compatible con diferentes dispositivos y protocolos.

5. Desarrollo de la solución

- **Plataforma IoT:** La plataforma IoT debe ofrecer una interfaz sencilla e intuitiva además de ser compatible con distintos dispositivos IoT. Asimismo, debe proporcionar un historial de los datos registrados así como la posibilidad de visualizarlos en tiempo real mediante diferentes gráficas y widgets.

5.2. Implementación de los nodos

5.2.1. Componentes utilizados

Como se ha detallado en el punto 4, el dispositivo final estará formado por la placa de desarrollo Heltec Wi-Fi LoRa 32 V2.1, el sensor de CO₂ MH-Z19B y el sensor de temperatura y humedad DHT22. Todos estos componentes aparecen en la siguiente figura:

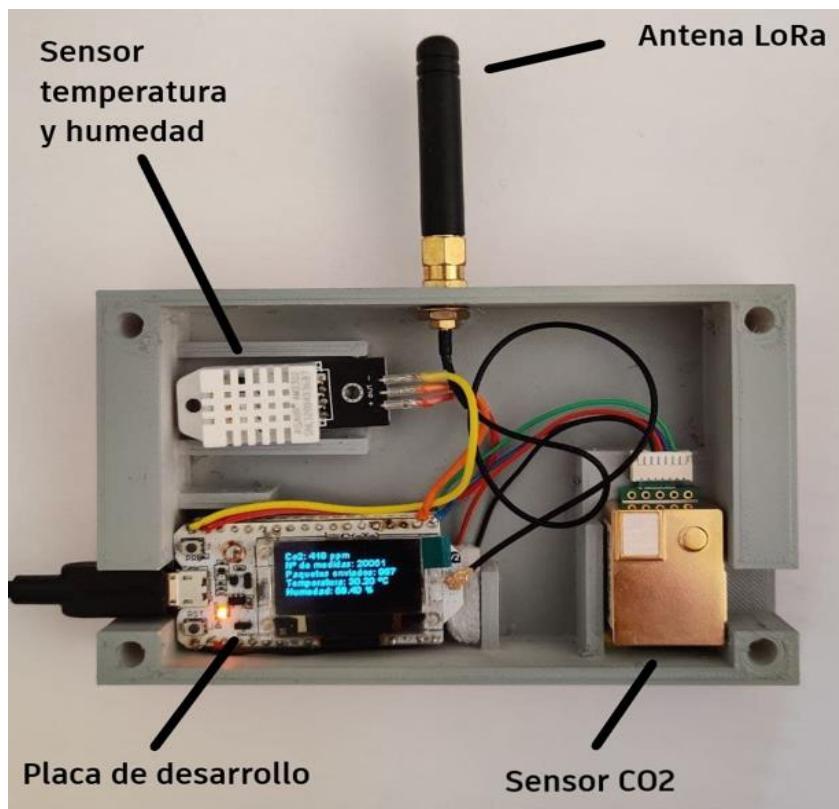


Figura 61. Esquema de los componentes del dispositivo (Fuente: Elaboración propia)

5. Desarrollo de la solución

➤ Sensor CO₂ MH-Z19B

Como se puede apreciar en la Figura 62, para conectar el sensor MH-Z19B a la placa de desarrollo, se utilizan unos pines ubicados en la parte izquierda del sensor. La conexión se realizará mediante un conector, como muestra la Figura 63.

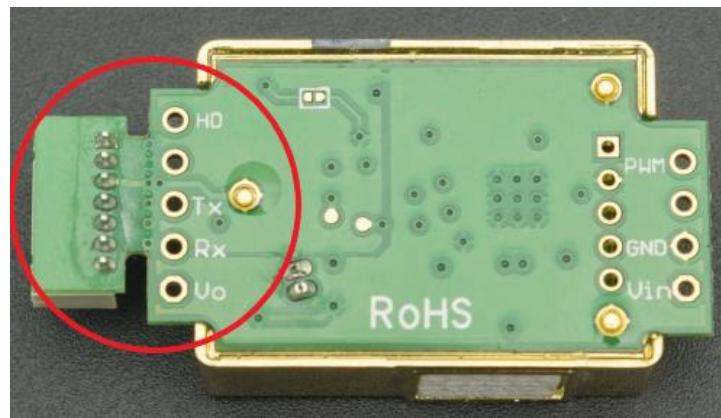


Figura 62. Pines del sensor MH-Z19B (Fuente: Elaboración propia)



Figura 63. Conector del sensor MH-Z19B (Fuente: Emariete)

5. Desarrollo de la solución

Pin	Color	Función	Conexión con la placa de desarrollo
-	Marrón	Sin conexión	-
-	Blanco	Sin conexión	-
GND	Negro	GND (Ground o tierra)	Pin GND
V_o	Rojo	Alimentación	Pin 5V
Rx	Azul	Recepción de datos	Pin 21
Tx	Verde	Envío de datos	Pin 22
-	Amarillo	Sin conexión	-

Tabla 14. Pines y conexiones del sensor MH-Z19B (Fuente: Emariete)

➤ Sensor de temperatura y humedad DHT22

La distribución de los pines del sensor DHT22 es la siguiente:



Figura 64. Pines del sensor DHT22 (Fuente: Vistronica)

5. Desarrollo de la solución

Pin	Función	Conexión con la placa de desarrollo
+	Alimentación	Pin 3V3
out	Transmisión datos	Pin 13
-	GND (Ground o tierra)	Pin GND

Tabla 15. Pines y conexiones del sensor DHT22 (Fuente: Elaboración propia)

➤ Esquema del conexionado de los sensores con la placa de desarrollo

Este esquema difiere ligeramente del original ya que en este TFG, se ha utilizado un conector para facilitar la conexión con la placa. No obstante, las conexiones con la placa son las mismas.

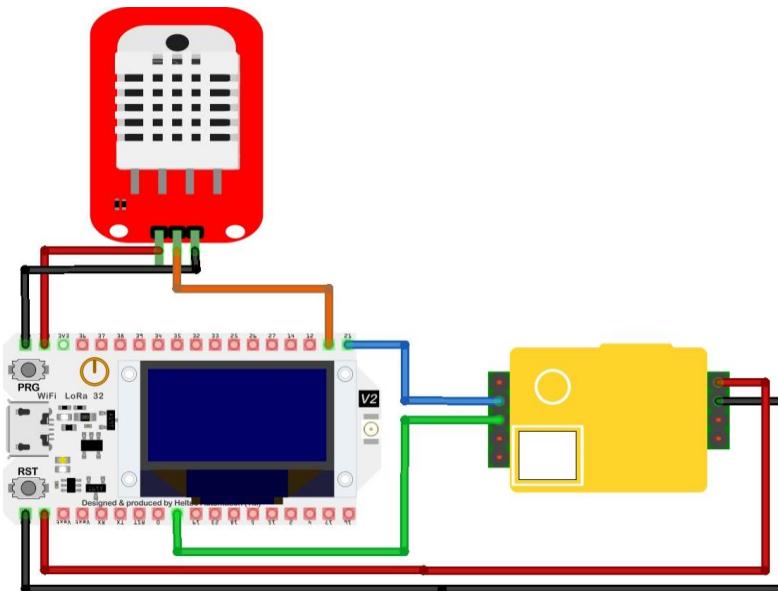


Figura 65. Esquema de las conexiones a la placa de desarrollo (Fuente: Elaboración propia)

5. Desarrollo de la solución

➤ Diagrama de pines de la placa Heltec Wi-Fi LoRa 32 V2.1

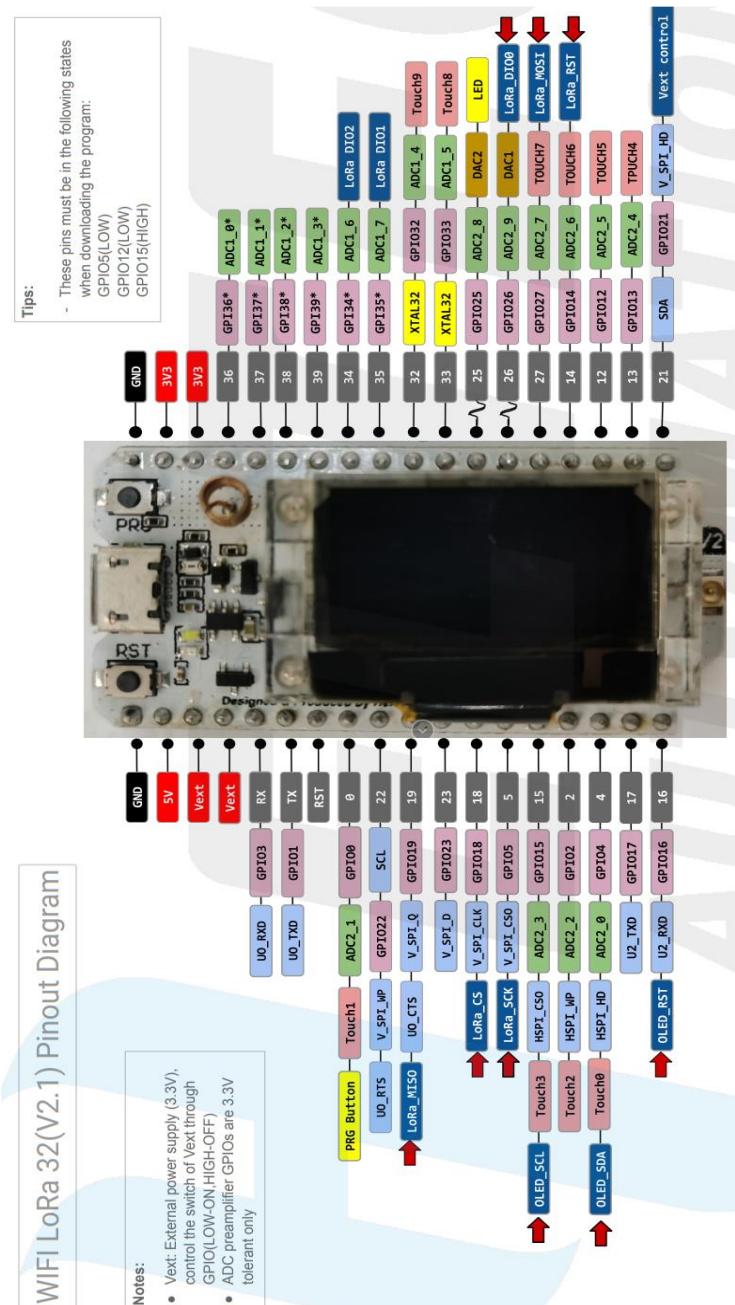


Figura 66. Diagrama de pines de la placa Heltec Wi-Fi LoRa 32 V2.1 (Fuente: Heltec)

5.2.2. Organigrama

Respecto al código de la placa de desarrollo, se adjunta en el anexo, en donde se detalla en qué consiste cada una de las funciones utilizadas así como cada uno de los métodos. No obstante, la lógica del código se resume en el siguiente organigrama:

Para facilitar la compresión del código, se ha elaborado el siguiente organigrama:

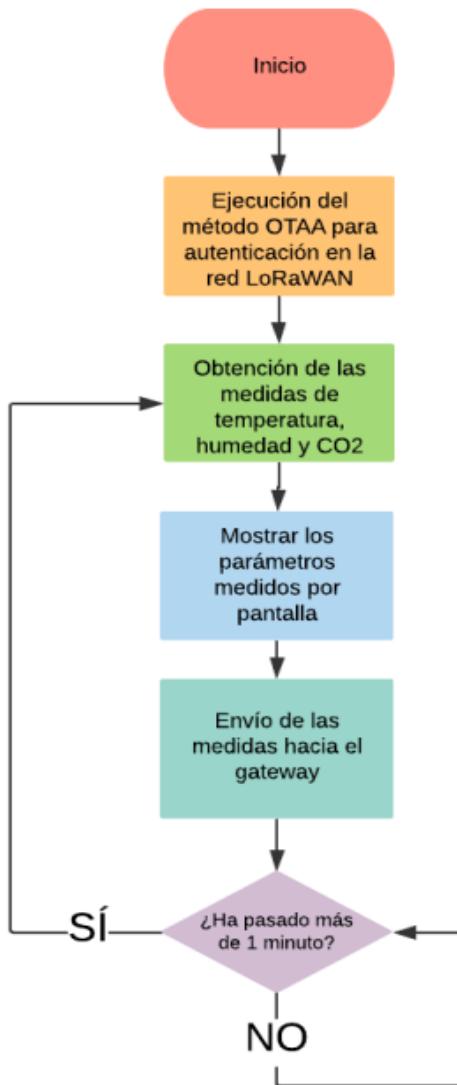


Figura 67. Organigrama del código (Fuente: Elaboración propia)

5. Desarrollo de la solución

5.2.3. Diseño de la carcasa 3D

Hasta ahora hemos trabajado con un prototipo inicial que se basaba en una protoboard para posibilitar la conexión de los sensores con la placa de desarrollo. No obstante, para la construcción del prototipo final, se ha diseñado una carcasa de manera que pueda proteger todos los componentes y mejore su estética. Para conseguirlo, se han elegido dos programas distintos.

Para la creación de los diseños se ha elegido TinkerCAD. Desarrollado por la empresa Autodesk, esta plataforma permite diseñar prototipos sencillos mediante figuras geométricas para su posterior impresión en 3D. Además, todas las piezas generadas pueden ser exportadas en formato STL (STereoLithography), el cual es un formato que pueden interpretar las impresoras 3D (M, 2020).



Figura 68. Logo de TinkerCAD (Fuente: TinkerCAD)

Una vez diseñado el prototipo, debemos exportarlo a un programa que sea capaz de comunicarse con la impresora 3D. Para ello se ha elegido el software PrusaSlicer. Se trata de un laminador compatible con multitud de impresoras 3D y archivos en formato 3MF, STL, OBJ o AMF, capaz de convertir diseños 3D en archivos de código G o G-code, el lenguaje que entienden las impresoras 3D.



Figura 69. Logo de PrusaSlicer (Fuente: PrusaSlicer)

5. Desarrollo de la solución

En primer lugar se ha diseñado la tapa y la carcasa del dispositivo que va a albergar tanto los sensores como la placa de desarrollo con la pantalla.

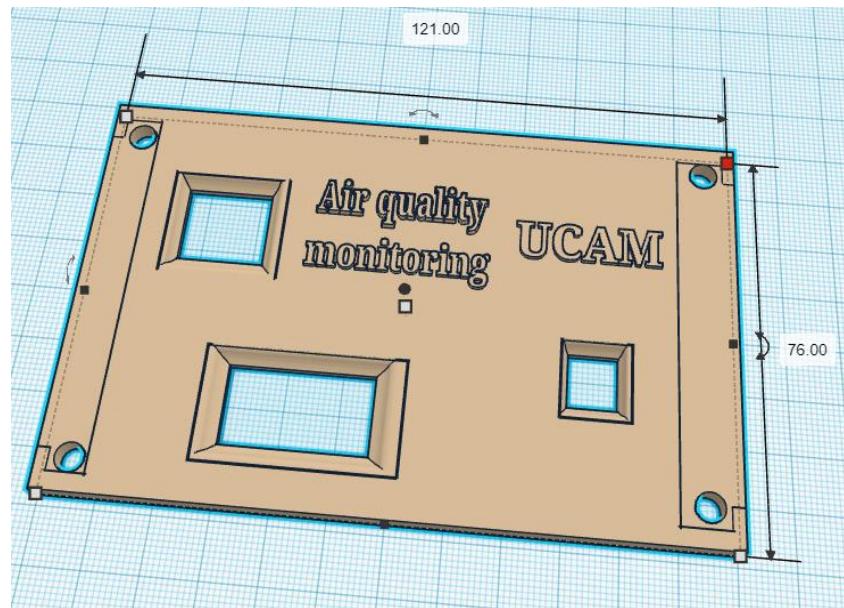


Figura 70. Diseño 3D de la tapa del dispositivo final (Fuente: Elaboración propia)

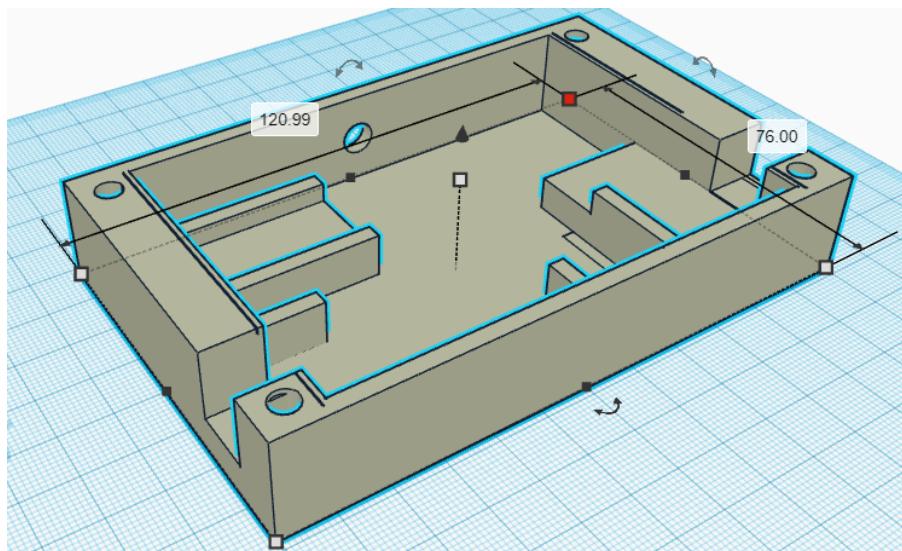


Figura 71. Diseño 3D de la carcasa del dispositivo final (Fuente: Elaboración propia)

5. Desarrollo de la solución

Una vez impresas ambas piezas, la parte interna del dispositivo con todos los componentes se muestra en la Figura 61, mientras que la parte externa quedaría de la siguiente manera:



Figura 72. Parte externa del dispositivo final impreso en 3D (Fuente: Elaboración propia)

5.3. Implementación del gateway

5.3.1. Puesta en marcha de la Raspberry Pi 4B y el concentrador

Como se detalló en el apartado 4.4.6 *Comparativa entre gateways LoRaWAN*, el gateway escogido para este trabajo es la Raspberry Pi 4B junto con el concentrador RAK2245. Antes de registrarlo en la red TTN, debemos modificar la memoria flash para con el objetivo de instalar un sistema operativo que contenga los archivos necesarios para que la Raspberry Pi 4B funcione como un gateway. Para ello, se ha escogido el sistema operativo BalenaOS.



Figura 73. Logo de BalenaOS (Fuente: Balena)

Desarrollado por la empresa que lleva el mismo nombre, BalenaOS es un sistema operativo basado en Linux que permite ejecutar contenedores de Docker en sistemas embebidos IoT. Entre los diferentes contenedores que podemos descargar, BalenaOS ha desarrollado uno que contiene los archivos necesarios para que la Raspberry Pi 4B junto con el concentrador RAK2245 funcionen como un gateway LoRaWAN multicanal compatible con la red TTN. Una de las principales características que ofrece BalenaOS comparado con otros sistemas operativos similares es el hecho de poder controlar y monitorizar el gateway de forma remota mediante un dashboard muy intuitivo sin necesidad de tener conocimientos previos de programación (BalenaOS, s.f.).

El procedimiento para la instalación del sistema operativo BalenaOS en la Raspberry Pi 4B es el siguiente:

5. Desarrollo de la solución

- 1) En primer lugar, debemos crearnos una cuenta en Balena Cloud desde el siguiente enlace: <https://dashboard.balena-cloud.com/signup>, donde tendremos que introducir nuestro email y una contraseña, como se puede apreciar en la Figura 74.

Email*

Password*

Send me the balena monthly newsletter
Get it once a month, e-mail is not shared with third parties.

Sign up

Figura 74. Creación de una cuenta en Balena Cloud (Fuente: Elaboración propia)

- 2) Una vez nos hayamos registrado, aparecerá una pantalla principal en donde tenemos que dirigirnos a la pestaña llamada “Applications” y posteriormente, “create application”. Nos aparecerá un menú para configurar el nombre de nuestra aplicación así como la versión de nuestra Raspberry Pi.

Use an existing application instead

Organization

g_pablo_del_arco's Organization

Application

tfg_covid

Default device type ?

Raspberry Pi 3

Application type [View docs](#)

Starter recommended

Figura 75. Menú de configuración de la aplicación en Balena Cloud (Fuente: Elaboración propia)

5. Desarrollo de la solución

- 3) A continuación, seleccionamos el botón “Add device” para registrar nuestra Raspberry Pi 4B. Nos aparecerá otro menú donde tendremos que indicar de nuevo el modelo de nuestra Raspberry Pi, la versión del sistema operativo así como el tipo de conexión a Internet, en nuestro caso debemos indicar “Ethernet only”.

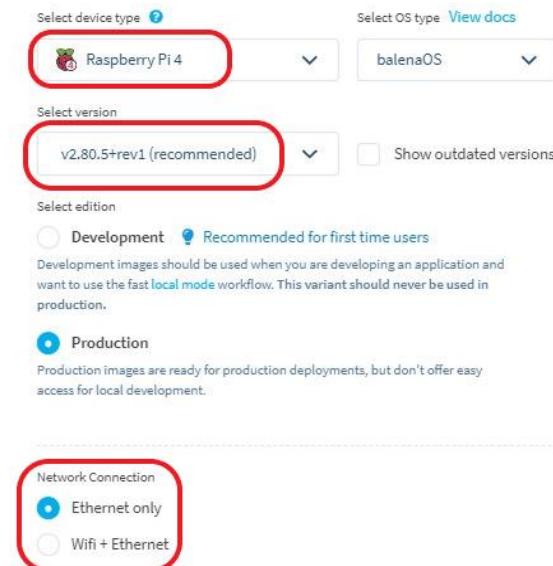


Figura 76. Menú para registrar un nuevo dispositivo (Fuente: Elaboración propia)

- 4) Cuando lo hayamos configurado todo, pulsamos en “Download BalenaOS” y se nos descargará una imagen personalizada con las opciones elegidas. Una vez descargado, usaremos el programa Etcher para instalar el sistema operativo en una microSD.



Figura 77. Menú para flashear la imagen de BalenaOS descargada (Fuente: Balena Etcher)

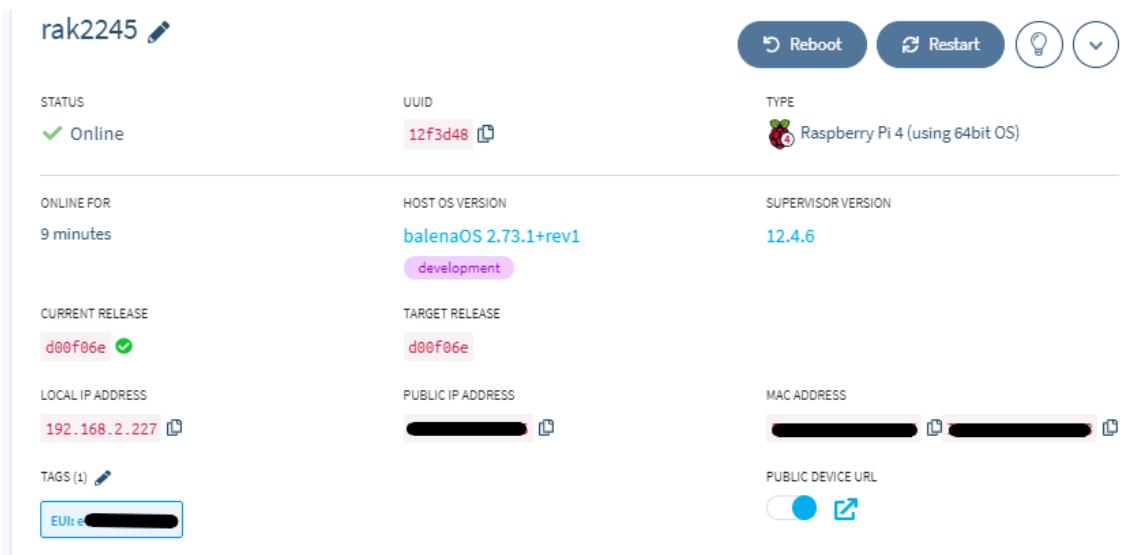
5. Desarrollo de la solución

- 5) Una vez se ha descargado, debemos insertar la tarjeta microSD en la Raspberry Pi, pero antes debemos acoplar el concentrador RAK2245, tal y como se muestra en la siguiente imagen:



Figura 78. Raspberry Pi 4B con el concentrador RAK2245 (Fuente: TTN)

- 6) Cuando hayamos insertado la microSD y conectado el cable de Ethernet a la Raspberry Pi, nos aparecerá “Online” en el Status del dashboard, lo cual significa que OS se ha flasheado satisfactoriamente. Desde el dashboard, podemos consultar tanto la IP pública como privada de nuestro gateway, la dirección MAC, la versión del sistema operativo, la temperatura o la memoria RAM, entre otras opciones.



Este dashboard es una interfaz web para gestionar un gateway. La cabecera indica "rak2245". Los botones principales son "Reboot", "Restart", "💡", y "▼".

STATUS	UUID	TYPE
✓ Online	12f3d48	Raspberry Pi 4 (using 64bit OS)

ONLINE FOR	HOST OS VERSION	SUPERVISOR VERSION
9 minutes	balenaOS 2.73.1+rev1 development	12.4.6

CURRENT RELEASE	TARGET RELEASE	MAC ADDRESS
d00f06e	d00f06e	[REDACTED]

LOCAL IP ADDRESS	PUBLIC IP ADDRESS	PUBLIC DEVICE URL
192.168.2.227	[REDACTED]	[REDACTED]

TAGS (1)
EUIb [REDACTED]

Figura 79. Dashboard que permite gestionar y monitorizar el gateway de forma remota (Fuente: Elaboración propia)

5. Desarrollo de la solución

Es importante anotar el código EUI que aparece en “TAGS”, ya que lo necesitaremos para registrar el gateway en la red TTN. El EUI es un identificador único formado por 8 bytes que permite diferenciar un gateway de otro. Por tanto, actuaría como una dirección MAC dentro de la red TTN.

5.3.2 Diseño del prototipo 3D

Al igual que ocurre con los nodos, debemos diseñar una carcasa que proteja el gateway. Si bien no estará colocado en exterior, es importante protegerlo para que nadie lo manipule. El diseño se ha hecho a medida para poder albergar la Raspberry Pi junto con el concentrador y además se ha ampliado la carcasa para que pueda albergar tanto la antena GPS como la antena de LoRa.

Para la impresión se han empleado los mismos programas que se utilizaron para imprimir las carcasas de los nodos. El diseño final ha quedado de la siguiente manera:



Figura 80. Diseño 3D de la tapa del gateway (Fuente: Elaboración propia)

5. Desarrollo de la solución

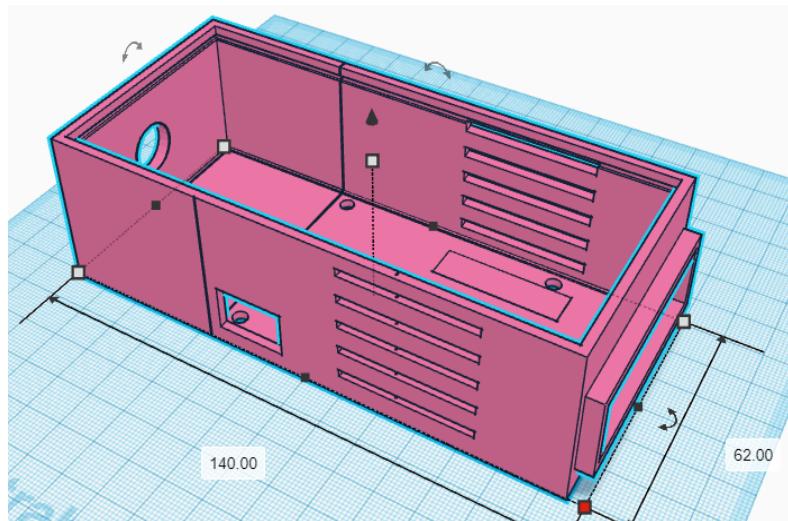


Figura 81. Diseño 3D de la carcasa del gateway (Fuente: Elaboración propia)

Una vez se hayan impreso las piezas 3D, el resultado final se muestra en las Figuras 82 y 83:



Figura 82. Parte externa del gateway impreso en 3D (Fuente: Elaboración propia)

5. Desarrollo de la solución



Figura 83. Parte interna del gateway impreso en 3D (Fuente: Elaboración propia)

5.4. Servidor TTN

Los desarrolladores de The Things Network han creado un servidor capaz de gestionar los datos transmitidos entre los nodos y gateways, además de poder registrar dichos dispositivos de forma gratuita. En este apartado, se explicará paso a paso cómo registrar tanto un gateway como un nodo en la red TTN.

5.4.1. Registro de los nodos LoRa en TTN

Lo primero que debemos hacer es acceder a este enlace <https://account.thethingsnetwork.org/register> para crearnos una cuenta en la página web de TTN. Una vez hemos accedido, debemos indicar nuestro nombre, email y contraseña. Cuando lo hayamos hecho, nos dirigimos hasta nuestro avatar y seleccionamos la opción “Console”, tal y como muestra la Figura 84:

5. Desarrollo de la solución



Figura 84. Página principal de TTN (Fuente: Elaboración propia)

A continuación, se nos mostrarán 2 opciones, “Applications” y “Gateways”. En esta ocasión empezaremos configurando el dispositivo final, así que seleccionamos “Applications”.

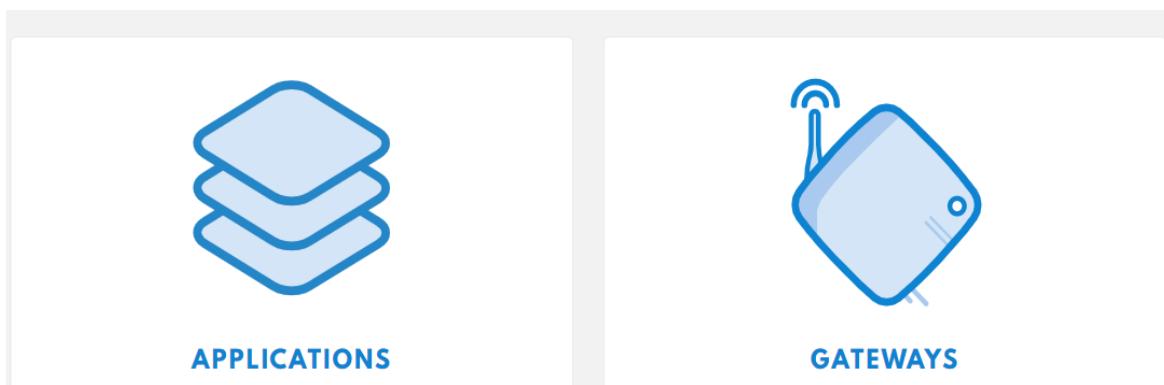


Figura 85. Página para registrar dispositivos en la red TTN (Fuente: Elaboración propia)

Antes de registrar un nodo en TTN, debemos crear una aplicación. Estas permiten gestionar distintos dispositivos y facilitan la gestión de los mismos. Para crear una, debemos introducir un nombre, una descripción (opcional) y el App EUI, aunque este último se genera de forma automática. Cuando hayamos completado los datos, pulsamos en “Add application”.

Sistema IoT de medición de CO₂ para prevención de COVID-19 basado en LoRaWAN

5. Desarrollo de la solución

The screenshot shows the 'ADD APPLICATION' form in the TTN interface. It includes fields for Application ID, Description, Application EUI, and Handler registration, each with a text input field and a green checkmark icon.

Application ID: The unique identifier of your application on the network.

Description: A human readable description of your new app. Example: Eg. My sensor network application.

Application EUI: An application EUI will be issued for The Things Network block for convenience, you can add your own in the application settings page. EUI issued by The Things Network.

Handler registration: Select the handler you want to register this application to. ttn-handler-eu.

Figura 86. Menú para crear una aplicación en TTN (Fuente: Elaboración propia)

A continuación, nos aparecerá un menú donde podremos gestionar los diferentes parámetros de nuestra aplicación. Para poder registrar un nodo, debemos pulsar la pestaña superior llamada “Devices”:

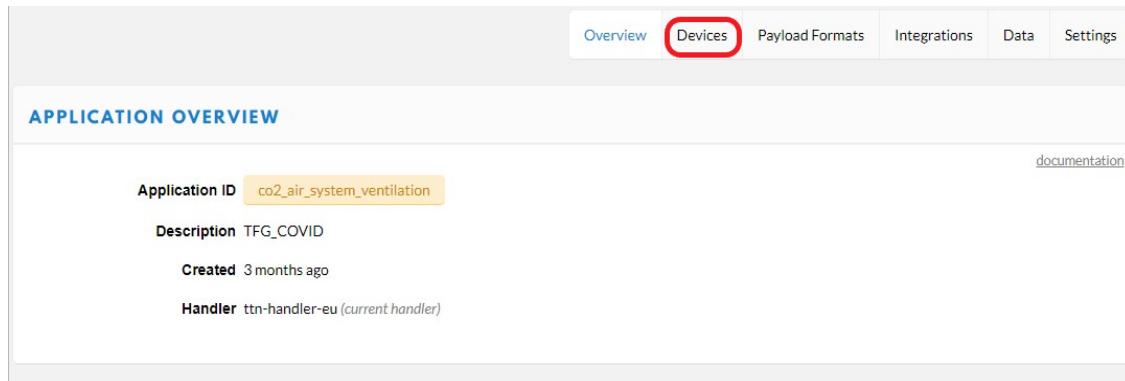


Figura 87. Menú principal para gestionar una aplicación en TTN (Fuente: Elaboración propia)

5. Desarrollo de la solución

Cuando hayamos accedido a la opción “Devices”, seleccionamos la opción “register device” y nos aparecerá el siguiente menú:

The screenshot shows a registration form for a device. At the top left is the title "REGISTER DEVICE". At the top right is a link "bulk import devices". Below the title, there are three main sections: "Device ID", "Device EUI", and "App Key". Each section has a label, a description, and an input field. The "Device ID" section has a note: "This is the unique identifier for the device in this app. The device ID will be immutable." The "Device EUI" section has a note: "The device EUI is the unique identifier for this device on the network. You can change the EUI later." The "App Key" section has a note: "The App Key will be used to secure the communication between your device and the network." At the bottom right of the form are two buttons: "Cancel" and "Register".

Figura 88. Menú para registrar un nodo en la red TTN (Fuente: Elaboración propia)

Los parámetros que debemos introducir son:

- **Device ID:** Se trata del nombre con el que vamos a identificar a nuestro nodo dentro de la aplicación.
- **Device EUI:** Identificador único para cada dispositivo dentro de la red TTN. Lo podemos escribir nosotros o bien podemos generarlo de forma aleatoria. Está formado por 8 bytes.
- **App Key:** Formado por 16 bytes, se trata de un código que permite asegurar la comunicación entre el nodo y la red TTN. Al igual que el Device EUI, lo podemos escribir nosotros o bien crear uno de forma aleatoria.

5. Desarrollo de la solución

Cuando hayamos completado todos los campos, pulsamos en el botón “Register” y ya habremos registrado nuestro nodo en TTN. La Figura 89 muestra el menú donde podemos gestionar los diferentes parámetros de nuestro nodo, como por ejemplo el nombre que le hemos asignado, el método de activación (OTAA en este caso), o las claves necesarias para conectarlo a la red TTN.

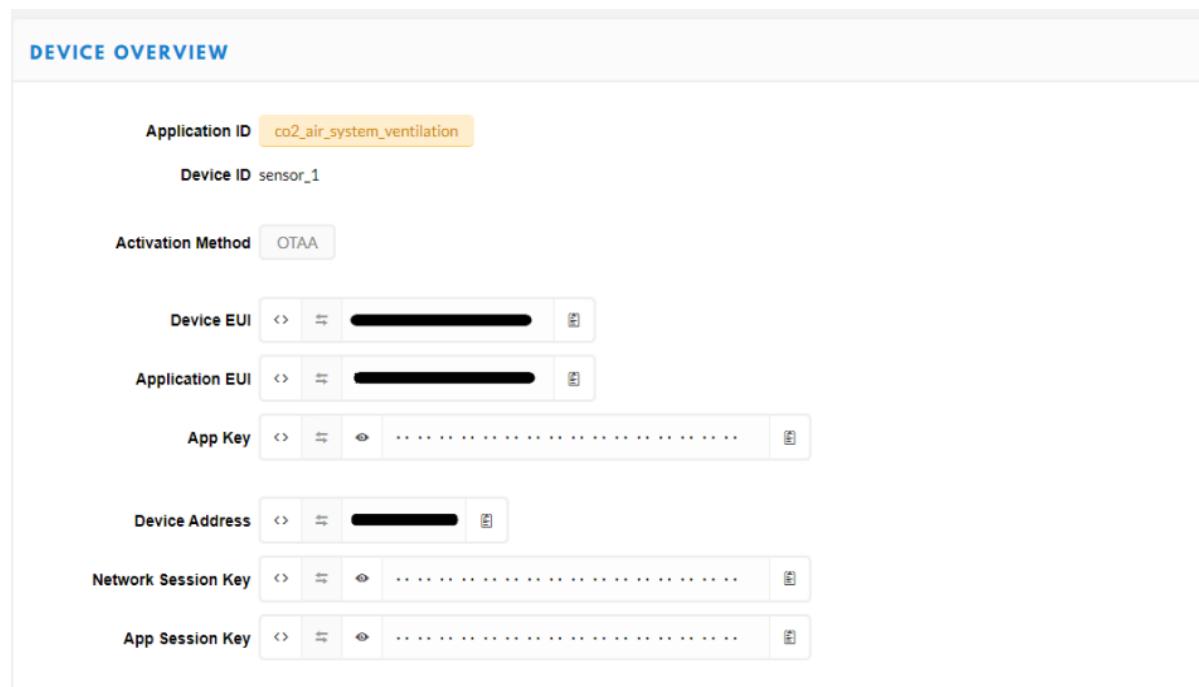


Figura 89. Menú para gestionar el nodo registrado en la red TTN (Fuente: Elaboración propia)

5. Desarrollo de la solución

Para que la conexión de nuestro nodo con la red TTN se haga efectiva, debemos introducir el Device EUI, Application EUI y App Key en el código que hemos analizado anteriormente, concretamente en esta parte del código:

```
// lsb (little-endian)
static const ul_t PROGMEM APPEUI[8] =
void os_getArtEui (ul_t* buf) {
    memcpy_P(buf, APPEUI, 8);
}

// lsb (little-endian)
static const ul_t PROGMEM DEVEUI[8] =
void os_getDevEui (ul_t* buf) {
    memcpy_P(buf, DEVEUI, 8);
}

// msb (big-endian)
static const ul_t PROGMEM APPKEY[16] =
void os_getDevKey (ul_t* buf) {
    memcpy_P(buf, APPKEY, 16);
}
```

Figura 90. Fragmento del código donde aparece el Device EUI, Application EUI y AppKey
(Fuente: Elaboración propia)

5.4.2. Registro del gateway en TTN

Para poder registrar el gateway, debemos seguir el siguiente proceso:

- 1) En primer lugar es necesario registrar en la web de TTN, no obstante, este paso ya se ha explicado en el punto anterior así que pasamos directamente al siguiente paso.
- 2) En segundo lugar, debemos entrar en el siguiente enlace <https://v2console.thethingsnetwork.org/gateways> y seleccionar la opción “register gateway”. Cuando lo hayamos hecho, veremos un menú como muestra la figura 91:

Sistema IoT de medición de CO₂ para prevención de COVID-19 basado en LoRaWAN

5. Desarrollo de la solución

REGISTER GATEWAY

Gateway EUI
The EUI of the gateway as read from the LoRa module
8 bytes

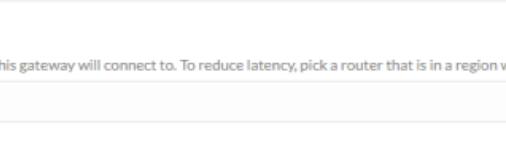
I'm using the legacy packet forwarder
Select this if you are using the legacy [Semtech packet forwarder](#).

Description
A human-readable description of the gateway

Frequency Plan
The [frequency plan](#) this gateway will use

Router
The router this gateway will connect to. To reduce latency, pick a router that is in a region which is close to the location of the gateway.

Location
The exact location of your gateway. This will be used if your gateway cannot determine its location by itself. Set a location by clicking on the map.



lat	0.0000000
lng	0.0000000
lat	0.0000000
lng	0.0000000

Figura 91. Menú para registrar el gateway en la red TTN (Fuente: Elaboración propia)

- **Gateway EUI:** Se trata del identificador del Gateway y se encuentra en el dashboard de Balena Cloud.
 - **Opción “I'm using the legacy packet forwarder”:** Activando esta opción, le estamos indicando a TTN que nuestro gateway, mediante el protocolo IP/UDP, reenvía hacia el servidor los paquetes de datos recibidos por la antena conectada al concentrador, además de emitir paquetes de datos que proceden del servidor hacia los nodos (The Things Network, 2018).
 - **Descripción:** Es opcional.
 - **Frequency Plan:** Se trata de la frecuencia a la que opera nuestro gateway. En este caso, debemos seleccionar “Europe 868 MHz”.

5. Desarrollo de la solución

- **Router:** Se trata del router al que se va a conectar nuestro gateway a través de Internet. Debemos elegir aquel que esté más cerca de la ubicación del gateway. En este caso elegimos “ttn-router-eu”.
- **Location:** Se trata de un mapa para indicar la ubicación de nuestro gateway. Es de gran utilidad ya que de esta manera, otras personas podrán conocer las zonas en las que hay cobertura LoRa.

- 1) A continuación, se nos mostrará un menú donde podemos ver todos los parámetros que hemos configurado así como el estado del gateway. Si el proceso se ha completado de forma satisfactoria, en la opción Status debería aparecer como “connected”, que significa que nuestro gateway está conectado a la red TTN. Además, podemos hacer pública otra tipo de información como por ejemplo el tipo de antena utilizada, la marca o el modelo de nuestro gateway.

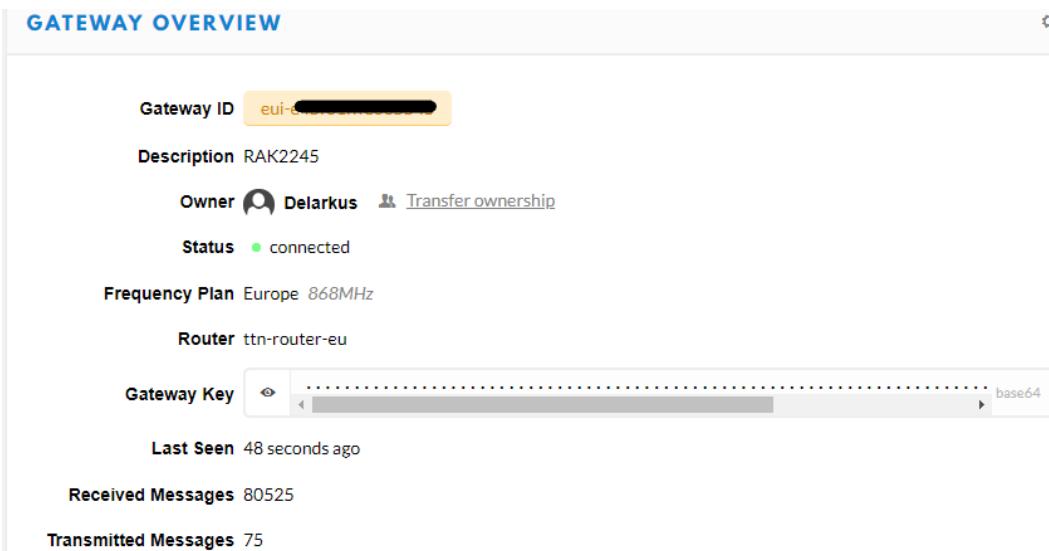


Figura 92. Menú para gestionar el gateway registrado en la red TTN (Fuente: Elaboración propia)

5. Desarrollo de la solución

5.5. Procesado de datos y plataformas IoT

TTN ofrece una sencilla interfaz que recoge todos los datos procedentes de los dispositivos finales. Los parámetros correspondientes a cada mensaje aparecen en la siguiente figura:

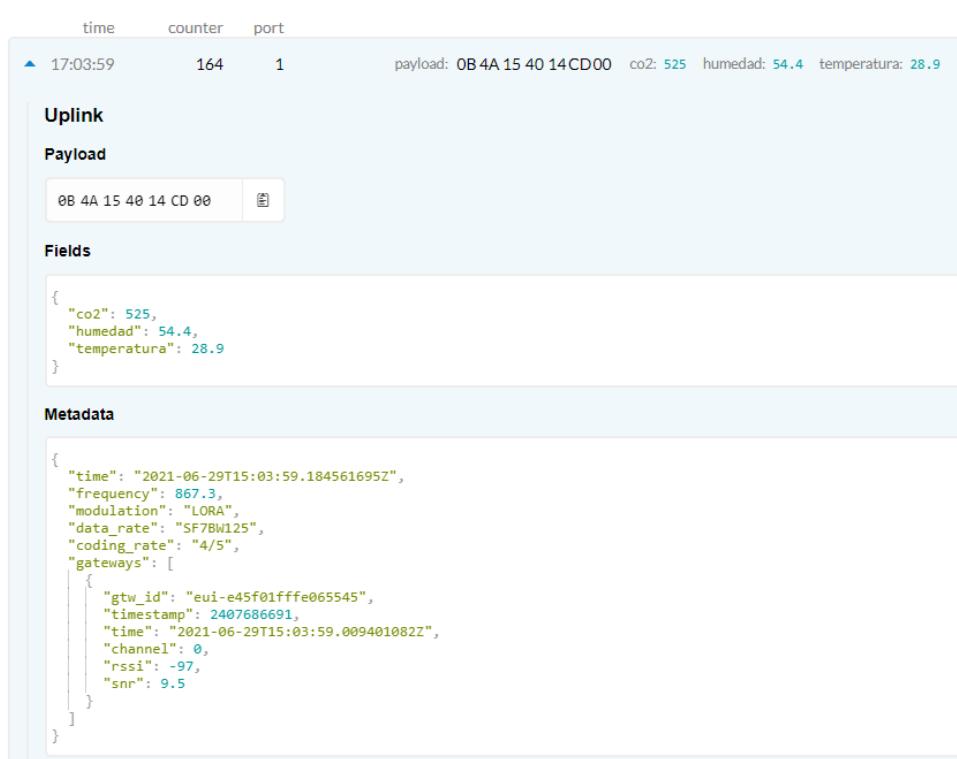


Figura 93. Interfaz visual que recoge los datos transmitidos por los nodos (Fuente: Elaboración propia)

- **Hora** de llegada del mensaje.
- **Contador**. Esto puede ser útil para saber si se ha perdido algún mensaje.
- **Payload**, aparece tanto en formato JSON como en hexadecimal.

5. Desarrollo de la solución

- **Metadata.** Entre los datos que forman los metadatos aparece el identificador EUI del gateway al que estamos enviando los datos, el canal por el que se está transmitiendo la información, la fuerza de la señal con la que el gateway recibe los datos (RSSI) y la relación señal a ruido (SNR), entre otros.

Los paquetes de datos en LoRaWAN se transmiten en formato hexadecimal en lugar de strings, con el objetivo de reducir el ancho de banda lo máximo posible y mejorar la eficiencia energética. Sin embargo, si deseamos traducir los paquetes de datos a un formato JSON para poder entenderlos, debemos hacer uso de una herramienta que TTN implementa de forma nativa para descodificar los datos (Sepúlveda, 2021).

Esta herramienta llamada “Payload Functions” convierte los mensajes que están en formato hexadecimal a JSON, como muestra la Figura 94:

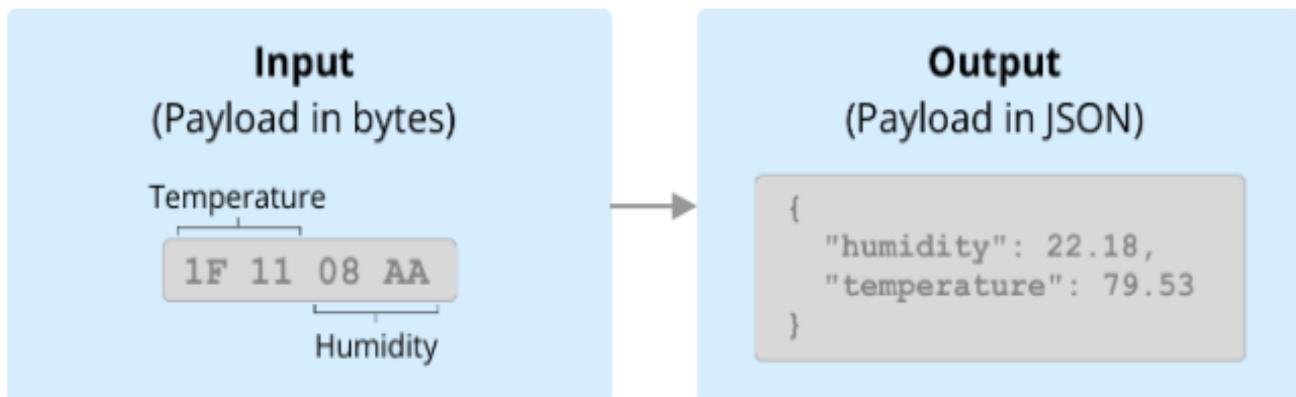


Figura 94. Conversión de mensajes en formato hexadecimal a formato JSON (Fuente: Ubidots)

Debido a que cada mensaje transmite datos distintos, debemos crear una función personalizada que permita decodificar los datos de temperatura, humedad y CO₂. Para ello, se ha desarrollado la siguiente función escrita en JavaScript:

5. Desarrollo de la solución

```
function Decoder(bytes, port) {  
    var decoded = {};  
  
    var temp= (bytes[0] << 8) | bytes[1]; // temperature °C  
    var hum=(bytes[2] << 8) | bytes[3]; // humidity %  
    var co2= (bytes[4]) + // CO2 ppm  
    ((bytes[5]) << 8)  
        + ((bytes[6]) << 16)  
        + ((bytes[7]) << 24);  
  
    // Decode integer to float  
    decoded.temperatura = temp/ 100;  
    decoded.humedad = hum/ 100;  
    decoded.co2 = co2/ 100;  
  
    return decoded;  
}
```

Por ejemplo, si recibimos el siguiente payload: **0B 4A 15 7C 3C D7 00** la función devolverá el mensaje que se muestra a continuación en formato JSON:

```
{  
  
    "co2": 551,  
  
    "humedad": 55,  
  
    "temperatura": 28.9  
  
}
```

Los 4 primeros dígitos (**0B 4A**) pertenecen a la temperatura, los 4 siguientes a la humedad (**15 7C**) mientras que los 3 últimos al CO₂ (**3C D7 00**).

5.5.2. Datacake

Tras probar diversos dashboard IoT, se ha elegido Datacake debido a que ofrece una interfaz muy intuitiva y completa, además de integrar diversas herramientas de personalización para la gestión de los datos procedentes de los sensores. Si bien Datacake ofrece una integración directa con los dispositivos LoRaWAN, se ha optado por integrarlos por MQTT ya que de esta

5. Desarrollo de la solución

manera es posible añadir todo tipo de funcionalidades extra como alertas de forma gratuita o la integración con otras aplicaciones mediante Node-RED (este proceso se explicará en el apartado 5.5.3.). El procedimiento que hay que seguir para poder utilizar el dashboard Datacake es el siguiente:

- 1) Si no estamos registrados en la página web de Datacake, debemos hacerlo mediante el siguiente enlace <https://app.datacake.de/signup>, donde tendremos que escribir nuestro nombre, apellidos, email y una contraseña. La figura 94 muestra la página de registro.

The screenshot shows a registration form with the following fields:

- First Name: Input field with placeholder "e.g. John".
- Last Name: Input field with placeholder "e.g. Doe".
- Email: Input field with placeholder "e.g. john.doe@example.com".
- Name of your first Workspace: Input field with placeholder "e.g. the name of your company or your name".
- Password: Input field with placeholder "Password".
- Confirm Password: Input field with placeholder "Confirm Password".

Figura 95. Página de registro en Datacake (Fuente: Elaboración propia)

- 2) A continuación, aparecerá la pantalla principal con una lista de opciones en la parte izquierda. Entre todas ellas, debemos elegir “Devices”, tal y como muestra la Figura 96:

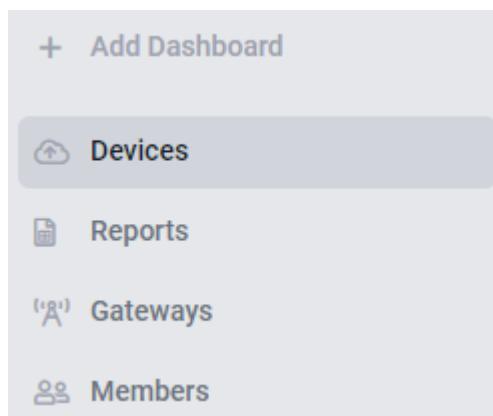


Figura 96. Menú de opciones del dashboard Datacake (Fuente: Elaboración propia)

5. Desarrollo de la solución

- 3) Cuando hayamos accedido al panel de dispositivos, seleccionamos el botón “Add device”.

Se nos abrirá un menú como el de la figura 97 donde tendremos que indicar la tecnología que estamos utilizando para la transmisión de los datos, el tipo de dispositivo así como su nombre. En nuestro caso tenemos que elegir “LoRaWAN”, en Datacake Product elegimos “New Product” y en Product Name escribimos el nombre del Producto, que se asemejaría a una aplicación que puede contener multitud de dispositivos.

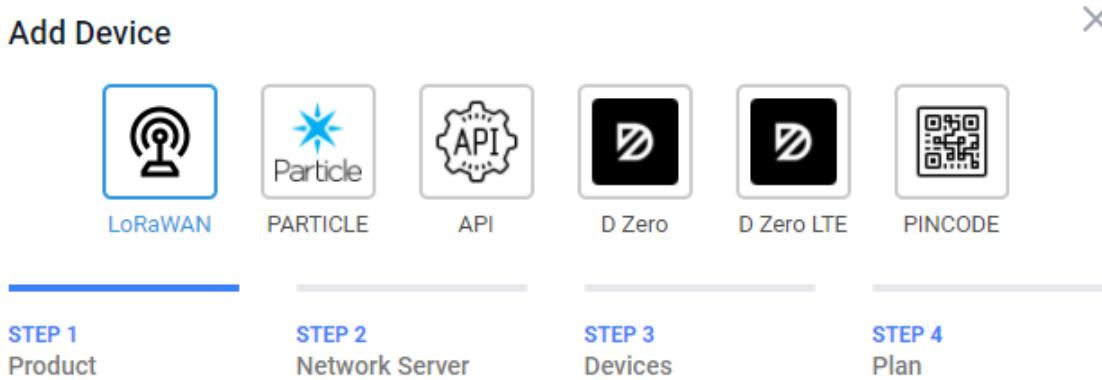


Figura 97. Menú de registro de un nuevo Producto en Datacake (Fuente: Elaboración propia)

Datacake Product

You can add devices to an existing product on Datacake, create a new empty product or start with one of the templates. Products allow you to share the same configuration (fields, dashboard and more) between devices.

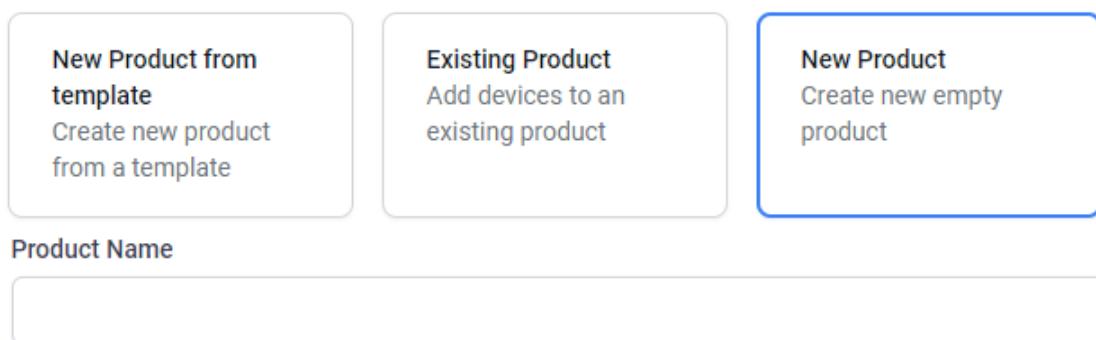


Figura 98. Menú de registro de un nuevo Producto en Datacake (Fuente: Elaboración propia)

5. Desarrollo de la solución

- 4) A continuación, debemos seleccionar el servidor de red, en nuestro caso es The Things Network V2:

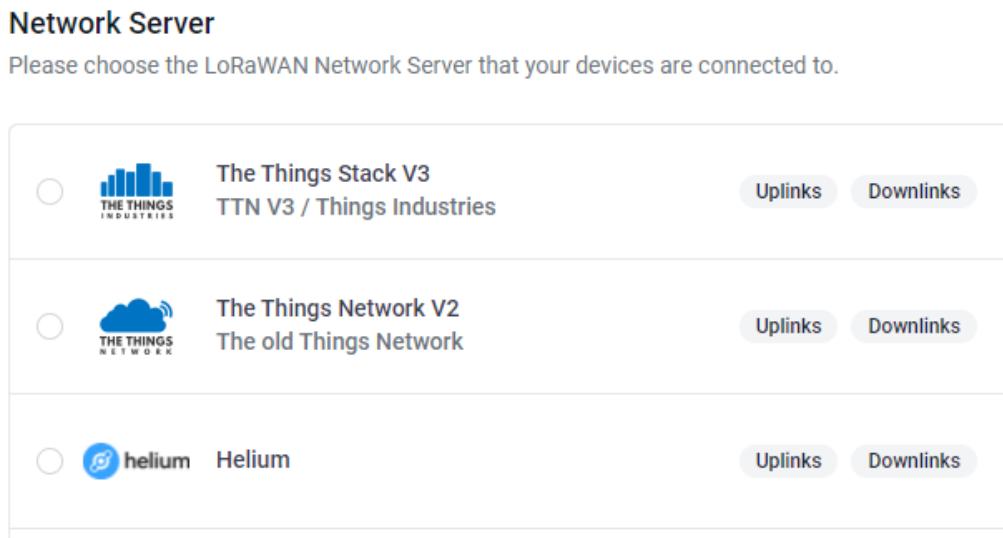


Figura 99. Menú para seleccionar el tipo de servidor de red (Fuente: Elaboración propia)

- 5) Por último, sólo nos falta indicar el identificador DevEUI, el cual se puede encontrar en el panel del dispositivo de TTN, así como el nombre con el que aparecerá en Datacake.

Add Devices

Enter one or more LoRaWAN Device EUIs and the names they will have on Datacake.

DEVEUI	NAME
DevEUI Please enter a DevEui	Name Please enter a name

Figura 100. Menú para registrar un nuevo dispositivo en Datacake (Fuente: Elaboración propia)

5. Desarrollo de la solución

De esta manera ya tendríamos nuestro dispositivo integrado en Datacake. Ahora es el momento de crear el dashboard para monitorizar los datos. Para ello, debemos dirigirnos a la opción llamada “Add Dashboard” en el panel izquierdo. Tras esto, seleccionamos el botón en el que parece “Add Widget” y aparecerá un menú con diferentes widgets para seleccionar, algunos de ellos se muestran en la siguiente figura:

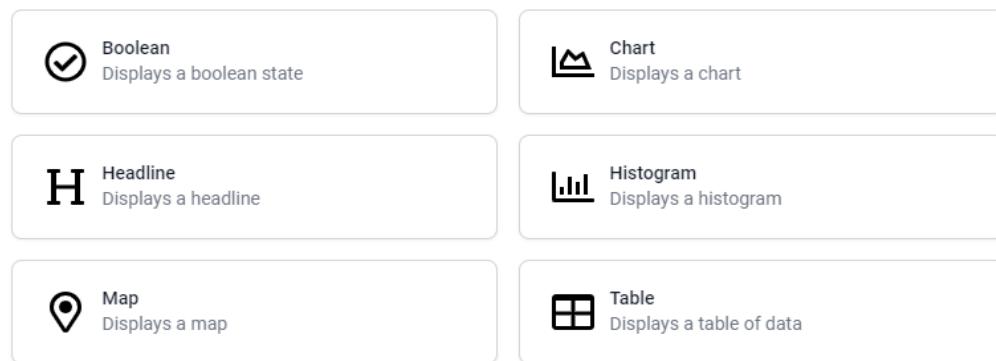


Figura 101. Widgets para visualizar los datos (Fuente: Elaboración propia)

Como los datos que se desea monitorizar son la temperatura, la humedad y el CO₂, es conveniente utilizar un widget que muestre tanto el valor numérico como un gráfico que refleje los valores anteriores para ver el histórico de las mediciones. De esta manera, el dashboard se ha desarrollado haciendo uso del widget “Chart” y “Value”, como se puede apreciar en las figuras 102, 103 y 104:

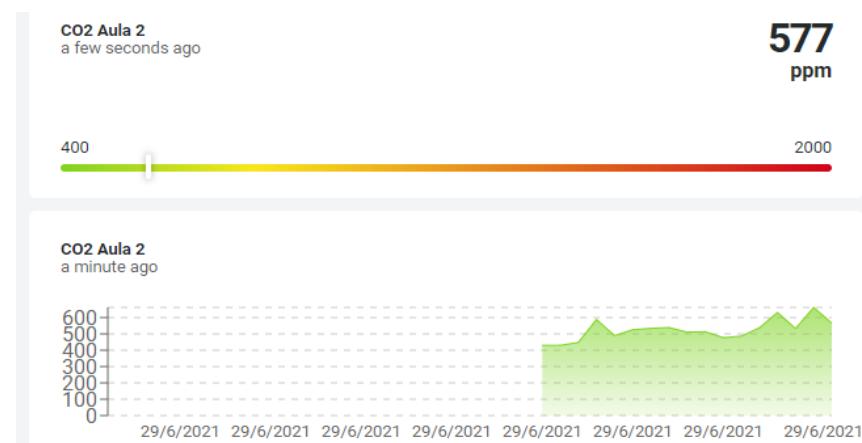


Figura 102. Histórico de mediciones de CO₂ en Datacake (Fuente: Elaboración propia)

Sistema IoT de medición de CO₂ para prevención de COVID-19 basado en LoRaWAN

5. Desarrollo de la solución

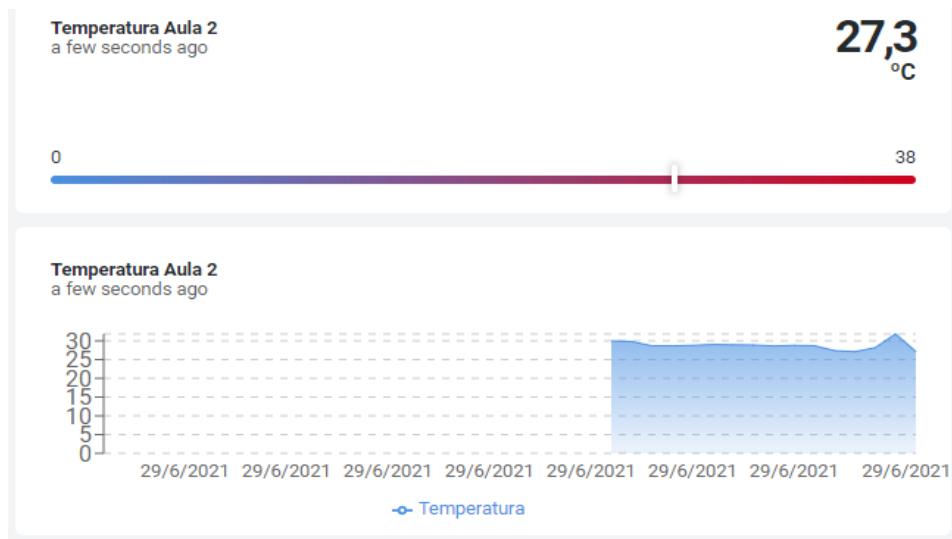


Figura 103. Histórico de mediciones de la temperatura en Datacake (Fuente: Elaboración propia)

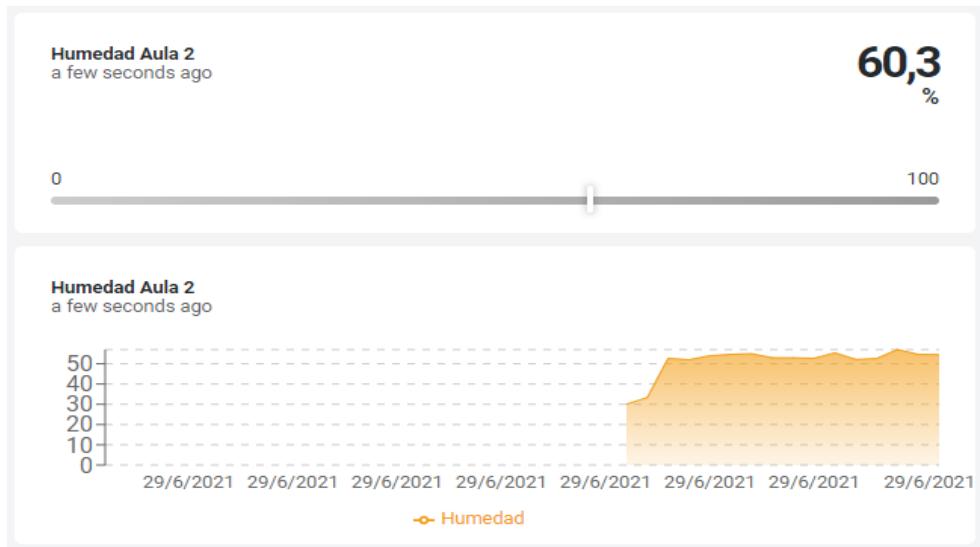


Figura 104. Histórico de mediciones de la humedad en Datacake (Fuente: Elaboración propia)

5.5.3. Node-RED

Como se ha comentado en el apartado anterior, para la gestión de los mensajes se ha recurrido a Node-RED. Entre otras funcionalidades, esta plataforma permite combinar distintos broker MQTT para el envío de mensajes entre ellos, además de ser compatible con diversos softwares como por ejemplo Telegram o Alexa. En este apartado se va a detallar cómo gestionar la recepción y el envío de mensajes a través de MQTT, y sobre la creación de alertas mediante la aplicación de mensajería Telegram con el objetivo de ser alertados cuando los niveles de CO₂ superen un determinado valor y sea necesario ventilar. Bien es cierto que Datacake permite notificar mediante SMS, sin embargo, esta opción es de pago.

➤ Gestión de mensajes mediante MQTT

Para poder visualizar los datos en la plataforma Datacake, se ha escogido el protocolo MQTT para el envío de los mensajes. Node-RED ofrece distintos nodos que permiten gestionar tanto el envío como la recepción de los datos procedentes de los sensores. En la Figura 105, se muestran los distintos nodos y flujos que se han utilizado:

5. Desarrollo de la solución

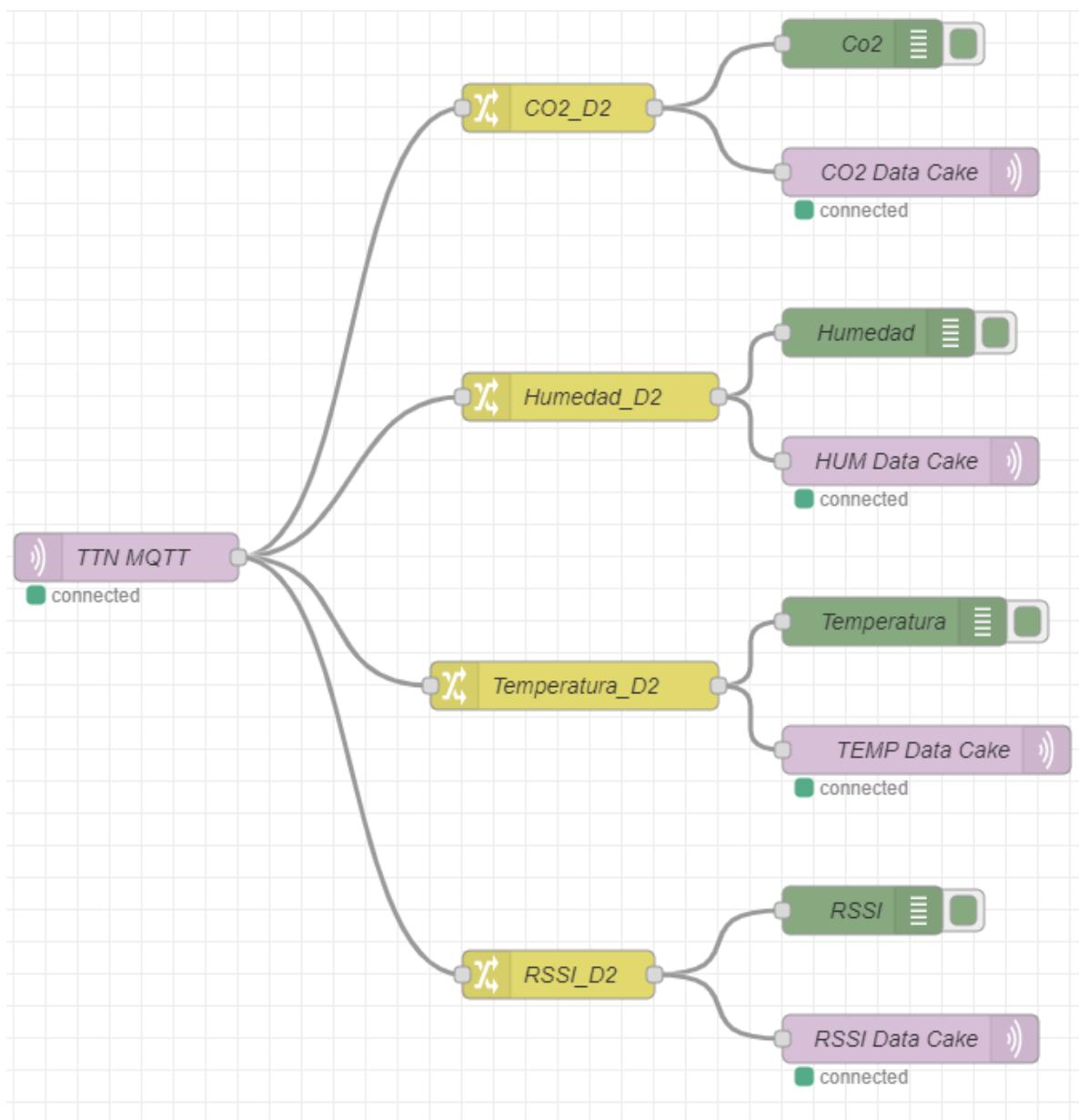


Figura 105. Estructura de nodos y flujos para la gestión de los datos (Fuente: Elaboración propia)

5. Desarrollo de la solución

El primer nodo llamado **TTN MQTT** sirve para establecer la conexión con el broker MQTT de TTN y permite además indicar el tópico al que deseamos suscribirnos. Se compone de los siguientes parámetros:

- **Server**: Se trata de un menú desplegable que permite seleccionar el broker del cual queremos recibir los datos. La URL del broker MQTT de TTN en Europa es **eu.thethings.network**, mientras que el nombre de usuario corresponde al ID de la aplicación y la contraseña al Application Access Key.
- **Topic**: Se ha de indicar el tópico al que queremos suscribirnos para, posteriormente, recibir los mensajes pertenecientes a este.
- **QoS**: Podemos configurar el nivel de calidad.
- **Output**: Mediante esta opción se indica a Node-RED el formato de los mensajes que se reciben.
- **Name**: Se ha de indicar el nombre del nodo con el que aparecerá en el panel de Node-RED.

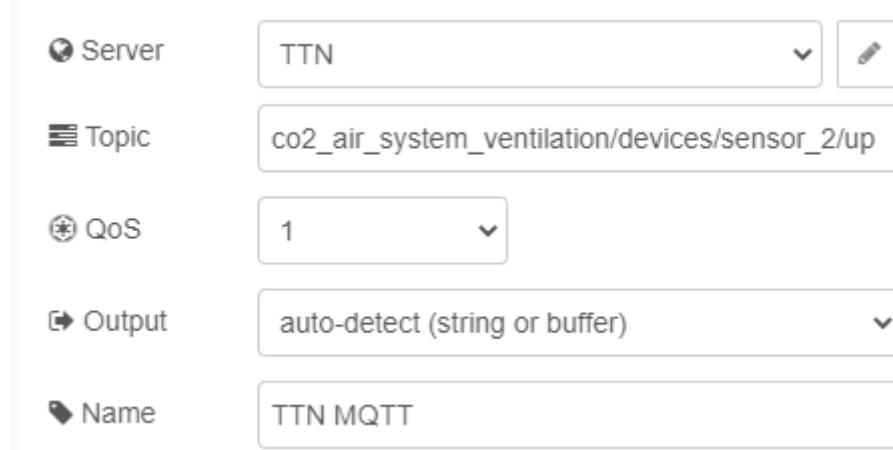


Figura 106. Menú del nodo MQTT TTN (Fuente: Elaboración propia)

5. Desarrollo de la solución

A continuación se encuentra el nodo amarillo del tipo **change**, el cual permite filtrar un mensaje que se incluye dentro de un texto en formato JSON. La siguiente figura muestra los parámetros que contiene este nodo:

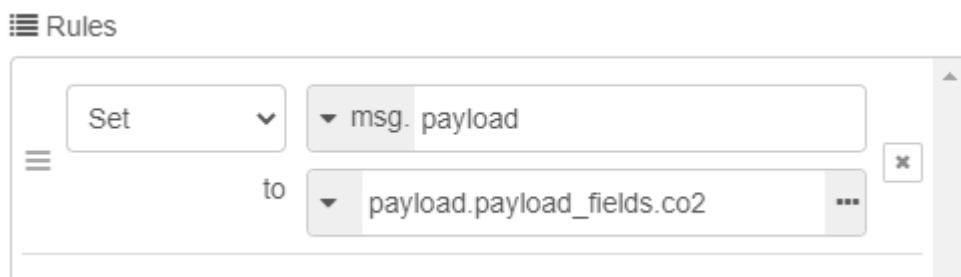


Figura 107. Menú del nodo **change** (Fuente: Elaboración propia)

Con respecto al parámetro superior, debemos indicar el tipo de mensaje mientras que en el segundo debemos escribir el dato que queremos filtrar, por ejemplo si deseamos filtrar la concentración de CO₂, debemos escribir **payload.payload_fields.co2**.

Por último, encontramos los nodos **debug** y **mqtt out**, los cuales aparecen de color verde y morado respectivamente. El primero de ellos permite mostrar por pantalla distintos valores de los nodos a los que esté conectado este nodo. Es de gran utilidad ya que podemos ver qué es lo que se está enviando hacia el broker de Datacake. El segundo nodo envía los distintos datos seleccionados hacia el broker MQTT de Datacake.

El nodo **debug** contiene dos principales parámetros, el primero se corresponde con el tipo de mensaje que se desea visualizar, mientras que el segundo es el nombre con el que se verá en el panel de Node-RED.

5. Desarrollo de la solución

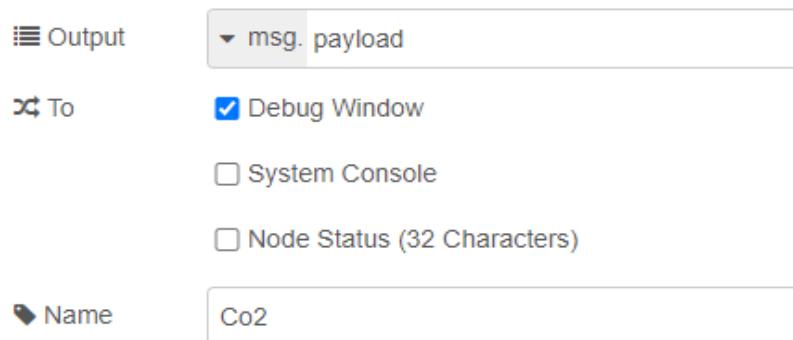


Figura 108. Menú del nodo debug (Fuente: Elaboración propia)

Respecto al nodo `mqtt out`, podemos encontrar los mismos parámetros que contiene el nodo `mqtt in`, aunque la única diferencia se encuentra en `Server`, ya que esta vez debemos indicar el broker MQTT al cual queremos enviar la información.

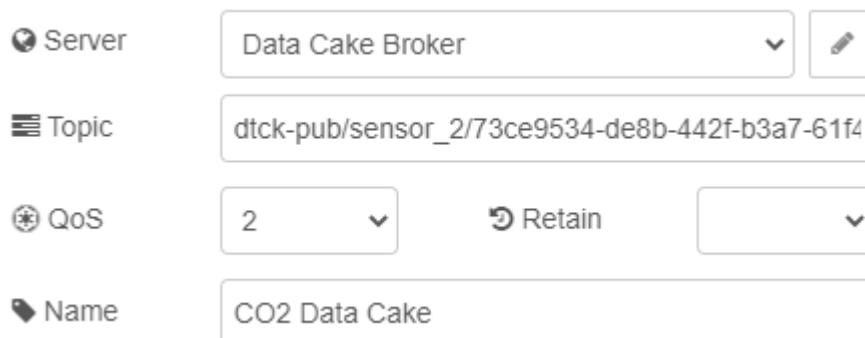


Figura 109. Menú del nodo `mqtt out` (Fuente: Elaboración propia)

➤ Alertas

Una vez que se ha detallado la gestión de los mensajes hacia el broker MQTT de Datacake, se propone configurar una serie de alertas para que cuando la concentración de CO₂ supere un determinado valor, se envíe un mensaje mediante la aplicación de Telegram. Como se comentó en el apartado 3. Relación entre la COVID-19 y el CO₂, a partir de 700 ppm se

5. Desarrollo de la solución

recomienda ventilar la sala, mientras que si la concentración supera los 1000 ppm, se debe ventilar la sala con urgencia.

Para ello, se ha desarrollado el siguiente programa:

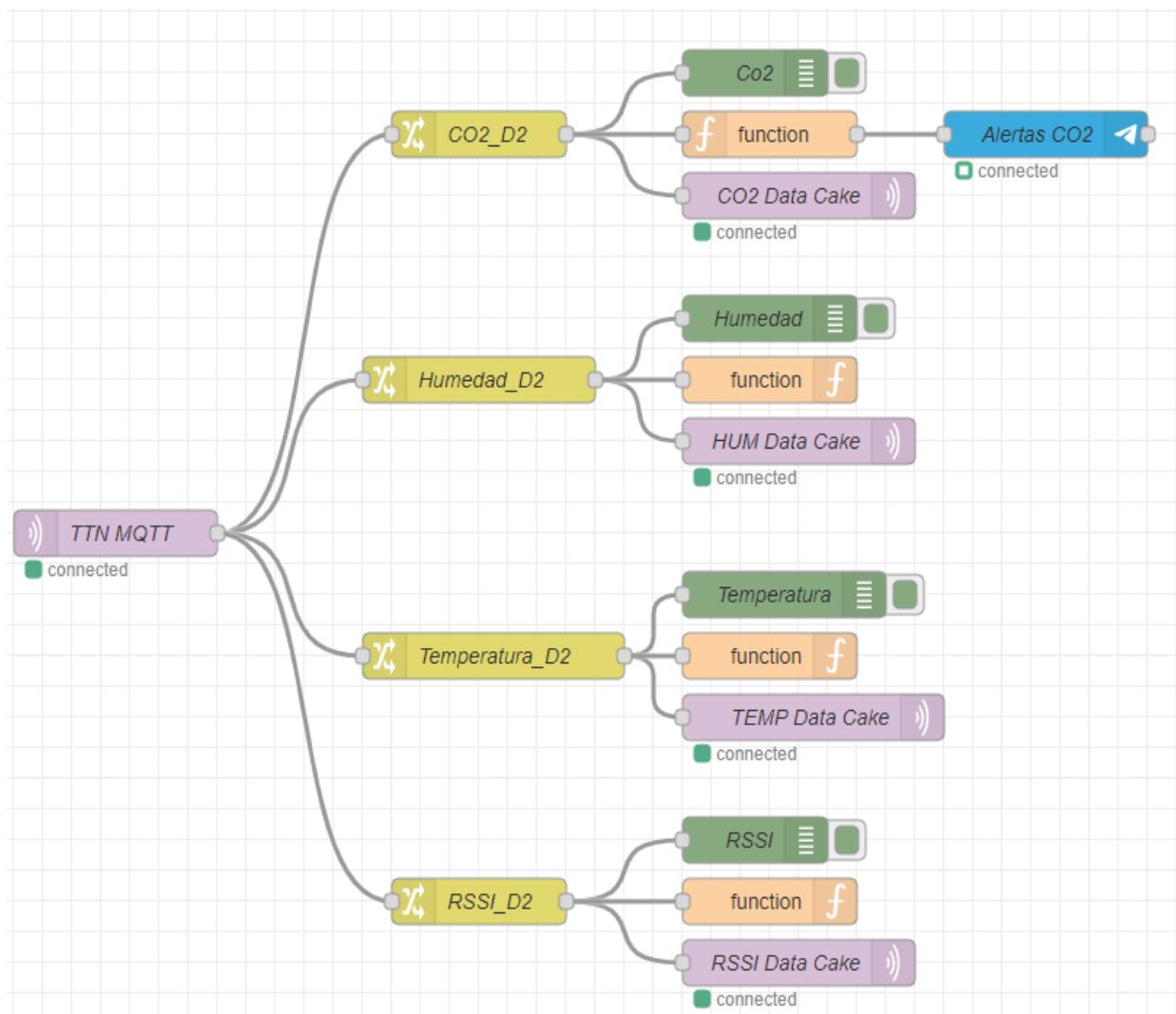


Figura 110. Estructura de nodos y flujos para la gestión de alertas y envío de mensajes
(Fuente: Elaboración propia)

5. Desarrollo de la solución

Para poder generar alertas en Telegram, se han añadido 2 nuevos nodos a la estructura anterior. Uno se corresponde con un nodo que permite el envío del mensaje hacia el bot de Telegram mientras que el otro se trata de un nodo del tipo `function` y es el que contiene el código que permite enviar las distintas alertas. El código utilizado es el siguiente:

```
flow.set("payload_co2", msg.payload);

if (msg.payload >= 700 && msg.payload < 1000) {

    msg.payload={

        "chatId": *****,
        "type":"message",
        "content": "Valor CO2 = " + msg.payload + " ppm. Se recomienda ventilar"
    }

    return msg;
}

} else if(msg.payload >= 1000) {

    msg.payload={

        "chatId": *****,
        "type":"message",
        "content": "Valor CO2 = " + msg.payload + " ppm. Alto riesgo, debes ventilar ahora!"
    }

    return msg;
}
```

5. Desarrollo de la solución

La lógica del programa se basa en que cuando el valor del CO2 supere los 700 ppm, se enviará una alerta a un bot de Telegram con el siguiente mensaje:

Valor CO2 = msg.payload ppm. Se recomienda ventilar

donde `msg.payload` corresponde con el valor de CO2 que haya medido en ese momento. Además, en caso de que se superen los 1000 ppm, se enviará la siguiente alerta:

Valor CO2 = msg.payload ppm. Alto riesgo, debes ventilar ahora!

El parámetro `chatId` es único para cada persona, por lo que si un nuevo usuario desea recibir las alertas, únicamente tendría que activar el bot cuyo enlace es `@alertas_tfg_bot` y posteriormente añadir su `chatId` en el código del nodo `function`, detallado anteriormente.

➤ Interacción con el bot de Telegram

Además de las alertas de Telegram, se ha decidido añadir también una nueva funcionalidad que consiste en que cualquier usuario que acceda al bot, pueda interactuar con él. Cuando queramos conocer la temperatura, el CO2, la humedad o la intensidad de señal, únicamente tendremos que enviar un mensaje al bot y este nos responderá con el dato solicitado en tiempo real.

Esta funcionalidad ha sido desarrollada mediante los siguientes nodos:

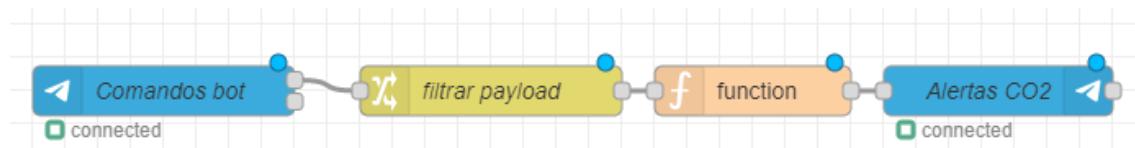


Figura 111. Estructura de nodos y flujos para interactuar con el bot de Telegram (Fuente: Elaboración propia)

5. Desarrollo de la solución

Los nodos que aparecen en la Figura 112 son los siguientes:

- **Nodo Comandos bot:** Permite recibir todos los mensajes enviados al bot en Telegram.
- **Nodo filtrar payload:** Permite seleccionar la parte del payload que corresponde con el mensaje enviado.
- **Nodo function:** Contiene la lógica que permite devolver un dato u otro en función del mensaje enviado. Este código es el que permite poder interactuar con el bot:

```
if (msg.payload == "temp" || msg.payload == "Temp") {  
  
    msg.payload={  
  
        "chatId": *****,  
  
        "type":"message",  
  
        "content": flow.get("payload_temp") + " °C"  
  
    }  
  
    return msg;  
  
} else if (msg.payload == "hum" || msg.payload == "Hum") {  
  
    msg.payload={  
  
        "chatId": *****,  
  
        "type":"message",  
  
        "content": flow.get("payload_hum") + " %"  
  
    }  
  
    return msg;  
  
} else if (msg.payload == "co2" || msg.payload == "Co2") {  
  
    msg.payload={
```

5. Desarrollo de la solución

```
        "chatId": *****,
        "type": "message",
        "content": flow.get("payload_co2") + " ppm"
    }

    return msg;
}

} else if (msg.payload == "rsssi" || msg.payload == "Rssi"){

msg.payload={

    "chatId": *****,
    "type": "message",
    "content": flow.get("payload_rssi") + " dBm"
}

return msg;
}
```

El funcionamiento del código consiste en que cuando el bot recibe por ejemplo la palabra co2 o Co2, devuelve la concentración de CO2 que se ha medido en ese momento. Para poder utilizar las medidas de CO2, temperatura, humedad y RSSI en esta función, estas se han almacenado previamente en variables globales que se incluyen en cada uno de los nodos naranjas llamados `function` conectados a cada uno de los nodos amarillos, como se muestra en la Figura 111. La sentencia para almacenar las medidas en variables globales es la siguiente:

```
flow.set("payload_co2", msg.payload);
```

- **Nodo Alertas CO2:** Por último, este nodo permite enviar al bot la medida que el usuario ha solicitado.

6. Resultados obtenidos

Una vez se han configurado los nodos, se ha puesto en marcha el gateway y se han configurado tanto la plataforma de IoT Datacake como Node-RED, es necesario realizar una serie de comprobaciones con el objetivo de analizar el funcionamiento de la solución propuesta en un escenario real.

6.1. Funcionamiento de los nodos

Las pruebas correspondientes a este punto consisten en comprobar si la pantalla de la placa de desarrollo muestra los valores de temperatura, humedad y CO₂ medidos en tiempo real, además de si los datos se envían satisfactoriamente hacia el gateway. La siguiente imagen muestra la pantalla del dispositivo en funcionamiento:



Figura 112. Pantalla del dispositivo final (Fuente: Elaboración propia)

6. Resultados obtenidos

Tal y como se puede apreciar en la Figura 113, la pantalla del dispositivo final muestra los siguientes valores:

- En la primera línea se muestra la concentración de CO₂ que hay en el aire cada 3 segundos.
- La segunda línea muestra un contador que aumenta en una unidad cada vez que se realiza una nueva medición.
- La tercera línea muestra la cantidad de paquetes que se han enviado de forma exitosa hacia el servidor, pasando por el gateway.
- En la cuarta línea aparece la temperatura medida cada 3 segundos.
- Por último, la quinta fila muestra la humedad medida por el sensor cada 3 segundos.

En vista de los resultados obtenidos en la pantalla del dispositivo, se puede comprobar que tanto las mediciones de los distintos parámetros como el envío de los datos hacia el servidor TTN se realiza de manera satisfactoria.

Respecto a los valores pertenecientes al número de medidas así como a la cantidad de paquetes enviados, aparecen por pantalla únicamente para realizar comprobaciones, no obstante, el dispositivo que se colocará en las aulas sólo mostrará los datos correspondientes al CO₂, temperatura y humedad.

6. Resultados obtenidos

6.2. Funcionamiento del dashboard IoT

Una vez se ha comprobado que los nodos funcionan satisfactoriamente, vamos a analizar a continuación el funcionamiento del dashboard Datacake. La Figura 114 y 115 muestran las distintas partes que forman el dashboard definitivo que permite visualizar toda la información procedente de los nodos:

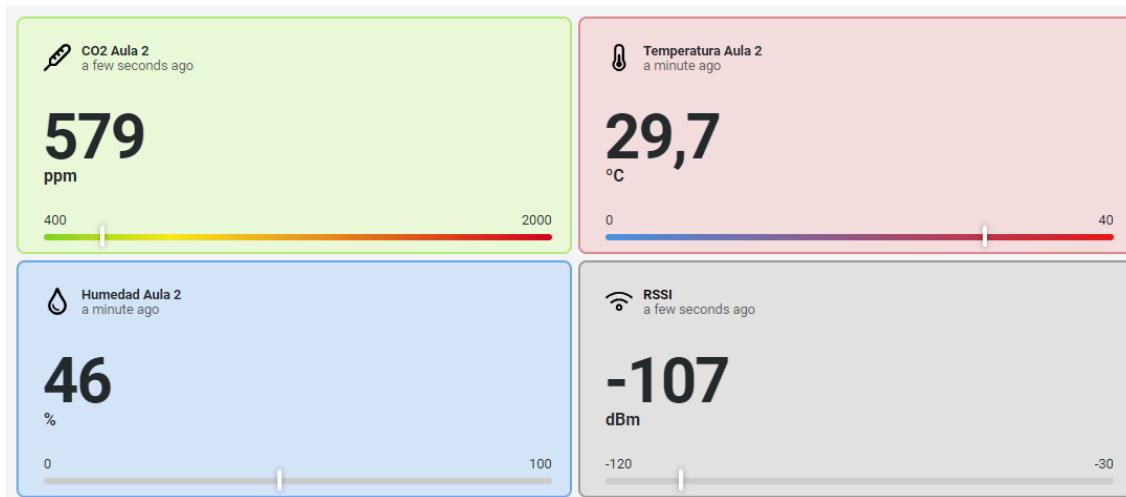


Figura 113. Parte del dashboard que numéricamente los valores medidas (Fuente: Elaboración propia)

La figura anterior muestra las medidas que corresponden al CO₂, temperatura, humedad e intensidad de señal con la que llegan al gateway. Todos los datos se muestran tanto de forma numérica como con una barra de nivel para poder visualizarlos de un modo más intuitivo.

6. Resultados obtenidos

La Figura 115 muestra la segunda parte del dashboard que contiene las distintas gráficas asociadas a cada medida. Cada una de ellas muestra un histórico de los datos correspondiente a un espacio temporal de un día, sin embargo, este puede ser configurado para que aparezcan únicamente los datos medidos en una hora, un día, un mes o un año. La primera gráfica de color verde pertenece al CO₂, la segunda, de color rojo, muestra las medidas de la temperatura, la tercera gráfica con un tono azulado muestra valores de la humedad mientras que la última, de color gris, muestra las intensidades de señal.

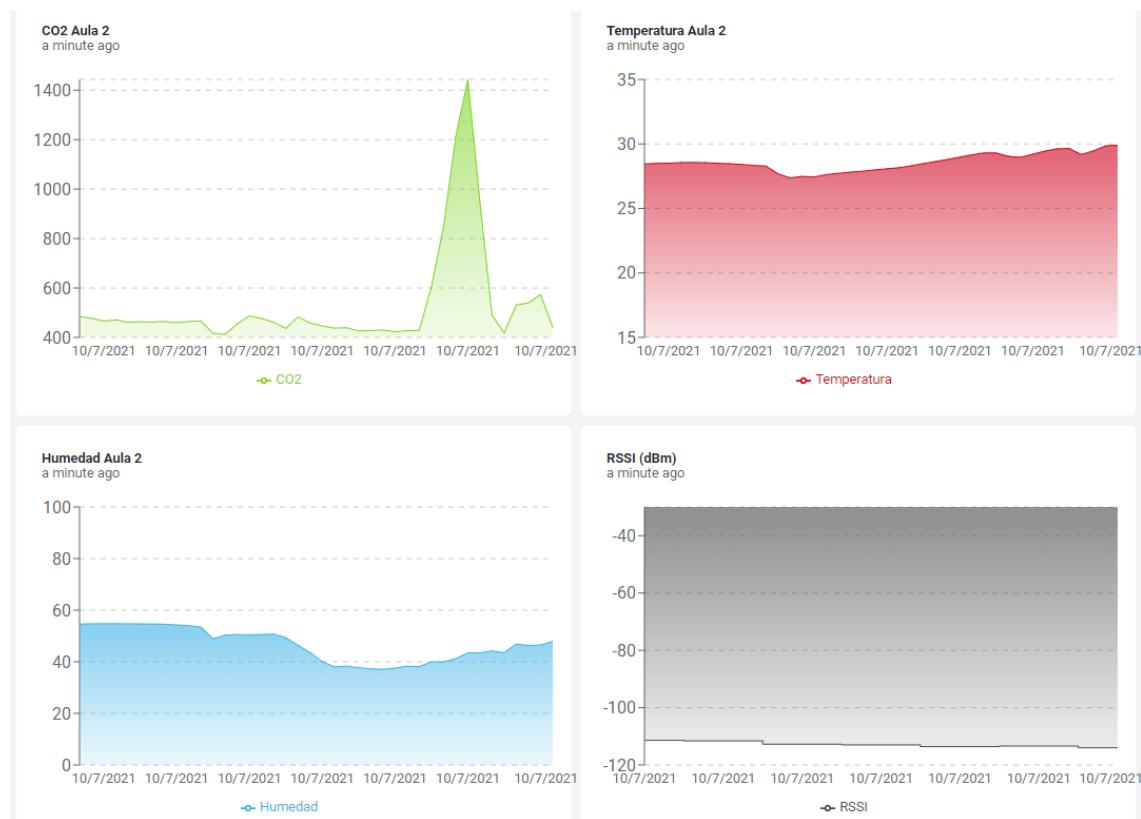


Figura 114. Parte del dashboard que las gráficas correspondientes a los valores medidos
(Fuente: Elaboración propia)

Como se puede apreciar en la figura anterior, los datos se reciben de manera correcta ya que estos quedan reflejados en las distintas gráficas. En caso de querer crear otros dashboard, se pueden diseñar otros distintos gracias a las herramientas que proporciona Datacake y sin coste adicional.

6. Resultados obtenidos

6.3. Funcionamiento de las alertas

Para poder verificar si las alertas se ejecutan de manera exitosa, se ha colocado el dispositivo en una habitación cerrada con varias personas hablando para que la concentración de CO₂ suba de manera intencionada.



Figura 115. Parte del dashboard que muestra el valor de la concentración de CO₂ (Fuente: Elaboración propia)

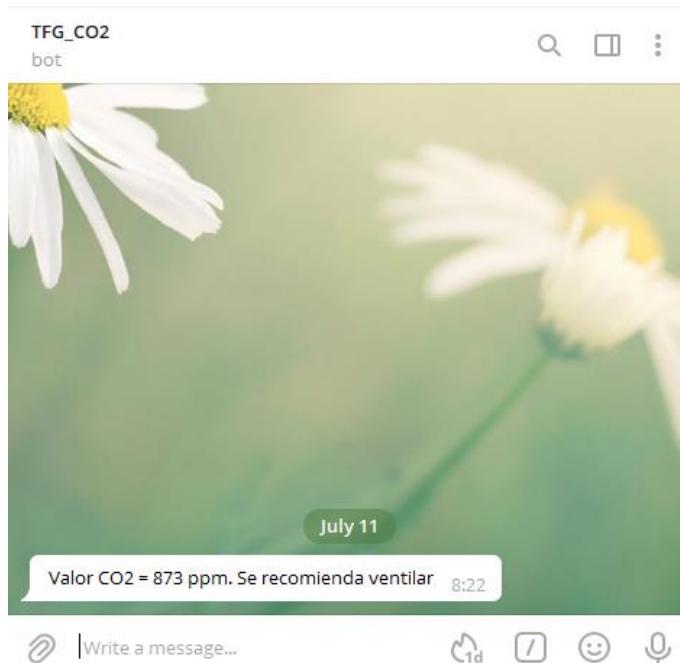


Figura 116. Bot de Telegram enviando una alerta (Fuente: Elaboración propia)

6. Resultados obtenidos

Por último, se va a comprobar si es el bot responde correctamente cuando recibe las palabras que se le han indicado previamente en Node-RED:



Figura 117. Bot de Telegram interactuando con el usuario (Fuente: Elaboración propia)

Como se puede observar en la Figura 118, el bot devuelve las medidas solicitadas de forma exitosa. Además, se ha programado de manera que reconozca las palabras tanto con la primera letra en mayúscula como en minúscula. Esto se debe a que cuando escribimos a través del teléfono móvil, por defecto, la primera letra siempre se escribe con mayúscula, mientras que si se escribe desde el ordenador, esto no sucede.

7. Presupuesto

7. Presupuesto

Con el objetivo de conocer el presupuesto final del proyecto, además del coste de los materiales, también se deben tener en cuenta las horas invertidas por el alumno, con una remuneración de 20€ / hora. En la siguiente tabla se desglosan los diferentes costes:

Concepto	Cantidad (unds)	Precio unitario	Total (€)
Personal ingeniería	500 horas	20€/hora	10000€
Placa de desarrollo Heltec Wi-Fi LoRa 32 V2.1	2	25,80€	51,6€
Antena 5 dBi	1	6,83€	6,83€
Sensor DHT22	2	6,495€	12,99€
Sensor MH-Z19B	2	35,99€	71,98€
Raspberry Pi 4B	1	59,90€	59,90€
Concentrador RAK2245	1	122.30€	122.30€
Fuente de alimentación	1	14€	14€
Cable micro USB	2	3,445€	6,89€
Cable Ethernet	1	9,85€	9,85€
Protoboard	3	2,763€	8,29€
Soldadura	2	10€	20€

Sistema IoT de medición de CO₂ para prevención de COVID-19 basado en LoRaWAN

7. Presupuesto

Material impresión 3D	19	1,69€	32,11€
Tarjeta MicroSD	1	10,99€	10,99€
TOTAL			10.427,73€

Tabla 16. Presupuesto del proyecto (Fuente: Elaboración propia)

8. Conclusiones y líneas futuras

8.1. Conclusiones

El presente TFG ha sido desarrollado como una solución para reducir el riesgo de contagio de COVID-19 mediante la medición, en tiempo real, de los niveles de concentración de CO₂ así como de la temperatura y humedad. Este sistema permite, además, alertar a los usuarios cuando se supera el nivel máximo admisible, disminuyendo así los posibles contagios debido a una mala ventilación.

En un principio se pensó desplegar los nodos únicamente en aulas de centros educativos, sin embargo, dada la versatilidad de la red LoRaWAN y el amplio rango de cobertura que ofrece un solo gateway, los nodos desarrollados en este proyecto pueden colocarse en cualquier recinto cerrado, como por ejemplo en oficinas, teatros, museos o zonas de restauración.

La estructura principal de este proyecto se basa en una red IoT de nodos que hacen uso de la tecnología LoRaWAN para el envío de datos hacia Internet. Cada minuto, estos nodos recogen los niveles de CO₂, temperatura y humedad, los cuales son transmitidos a un gateway de LoRaWAN. Posteriormente, estos datos son redirigidos hacia el servidor TTN donde se procesan y, mediante el protocolo MQTT, se envían a una plataforma IoT que ofrece diversas opciones para su visualización.

Haciendo uso de la aplicación de mensajería instantánea Telegram, se ha desarrollado un bot que permite enviar alertas cada vez que el nivel de CO₂ supera los 700 ppm, valor a partir del cual el riesgo de contagio de COVID-19 aumenta. Además de las alertas, se ha desarrollado también una funcionalidad extra que consiste en la interacción con el bot mediante mensajes de texto. Cada vez que el usuario envía al bot una palabra preprogramada, este devuelve la medida correspondiente en ese momento de la temperatura, CO₂, humedad o intensidad de señal. Se trata, por tanto, de una característica muy útil, especialmente cuando no se tiene acceso al dashboard IoT y se desea conocer la concentración de CO₂ u otro parámetro en tiempo real.

8.2. Líneas futuras

Como líneas futuras de cara a una mejora de este proyecto, se han propuesto las siguientes opciones:

- Implementación de una batería conectada a los nodos para no depender de la red eléctrica y aprovechar así las ventajas que ofrecen las tecnologías LPWAN, siendo el bajo consumo uno de sus puntos fuertes.
- Además de una batería, se podría implementar una placa solar en caso de que el sensor se colocara en una zona donde hubiera luz solar directa.
- Instalación de una antena de mayor ganancia que permita aumentar el rango de señal. De este modo, con un solo gateway, se podría ampliar la cobertura LoRa a un mayor número de edificios.
- Realizar un análisis de cobertura tomando medidas para determinar el mejor emplazamiento donde ubicar el gateway.
- Implementación de 3 LEDs (verde, amarillo y rojo) para conocer de un vistazo el nivel de CO₂ en la sala en tiempo real, así como una pantalla de mayor tamaño.
- Añadir otro tipo de sensor con el objetivo de medir distintos parámetros como por ejemplo un sensor de humo o un sensor que permita medir la contaminación acústica en las ciudades.

9. Índice de figuras

Figura 1. Conexiones totales de dispositivos inteligentes desde 2010	8
Figura 2. Arquitectura de un sistema IoT dividido en 5 capas	10
Figura 3. Concentración de partículas del virus SARS-CoV-2 en diferentes escenarios	12
Figura 4. Nivel de riesgo en función de la concentración de CO ₂	15
Figura 5. Comparativa entre las tecnologías de comunicación inalámbrica	17
Figura 6. Logo comercial de LTE-M	19
Figura 7. Bandas en las que opera NB-IoT	20
Figura 8. Logo comercial de NB-IoT	21
Figura 9. Mapa de cobertura de LTE-M y NB-IoT	22
Figura 10. Pila de protocolos de LoRaWAN	23
Figura 11. Estructura de la red LoRaWAN	24
Figura 12. Chirp de espectro ensanchado	26
Figura 13. Modulación DSSS	27
Figura 14. Señal modulada mediante DSSS	28
Figura 15. Mensaje con sus réplicas	32
Figura 16. Arquitectura SigFox	33
Figura 17. Módulo ESP32-WROOM-32U	37
Figura 18. ESP32-DevKitC	39
Figura 19. Node-MCU-32S	39
Figura 20. ESP32-CAM	40
Figura 21. Heltec Wi-Fi LoRa 32 V2.1	40
Figura 22. TTGO T-Beam V2.1	41
Figura 23. MH-Z19B	42

9. Índice de figuras

Figura 24. Espectro de absorción infrarroja del CO ₂	43
Figura 25. Parte interna del sensor MH-Z19B.....	43
Figura 26. MQ-135.....	45
Figura 27. Sensibilidad del sensor MQ-135	47
Figura 28. DHT11	48
Figura 29. Trama que contiene la información de temperatura y humedad.....	49
Figura 30. Esquema de una infraestructura IoT	50
Figura 31. RAK2245 Pi HAT	53
Figura 32. Componentes de la Raspberry Pi 4B	54
Figura 33. MikroTik wAP LR8 kit.....	55
Figura 34. Wirnet iStation	56
Figura 35. LIG16 Indoor Gateway.....	56
Figura 36. Logo del protocolo AMQP	59
Figura 37. Arquitectura AMQP	60
Figura 38. Comparativa entre el protocolo UDP y TCP	62
Figura 39. Arquitecturas de HTTP y CoAP	63
Figura 40. Transmisión confiable de mensajes	64
Figura 41. Transmisión no confiable de mensajes	64
Figura 42. Arquitectura MQTT	65
Figura 43. Diferentes niveles de tópicos	67
Figura 44. Proceso para recibir mensajes de un tópico concreto	69
Figura 45. Niveles de calidad de servicio (QoS)	70
Figura 46. Arquitectura LoRaWAN.....	73
Figura 47. Distribución de gateways LoRaWAN por todo el mundo	74

9. Índice de figuras

Figura 48. Logo de Datacake.....	75
Figura 49. Estructura de un Product de Datacake.....	76
Figura 50. Widget que ofrece el dashboard Datacake	78
Figura 51. Logo de Ubidots.....	78
Figura 52. Marcas compatibles con Ubidots	79
Figura 53. Widgets que ofrece Ubidots	80
Figura 54. Arquitectura de ThingSpeak.....	81
Figura 55. Ejemplo de un dashboard creado mediante ThingSpeak	82
Figura 56. Partes que componen un canal de ThingSpeak.....	83
Figura 57. Interfaz gráfica de Node-RED	85
Figura 58. Node-RED	87
Figura 59. Flujo de Node-RED.....	87
Figura 60. Pestaña de debug en Node-RED.....	88
Figura 61. Esquema de los componentes del dispositivo.....	91
Figura 62. Pines del sensor MH-Z19B	92
Figura 63. Conector del sensor MH-Z19B.....	92
Figura 64. Pines del sensor DHT22	93
Figura 65. Esquema de las conexiones a la placa de desarrollo.....	94
Figura 66. Diagrama de pines de la placa Heltec Wi-Fi LoRa 32 V2.1	95
Figura 67. Organigrama del código.....	96
Figura 68. Logo de TinkerCAD.....	97
Figura 69. Logo de PrusaSlicer.....	97
Figura 70. Diseño 3D de la tapa del dispositivo final.....	98
Figura 71. Diseño 3D de la carcasa del dispositivo final	98

9. Índice de figuras

Figura 72. Parte externa del dispositivo final impreso en 3D.....	99
Figura 73. Logo de BalenaOS	100
Figura 74. Creación de una cuenta en Balena Cloud.....	101
Figura 75. Menú de configuración de la aplicación en Balena Cloud	101
Figura 76. Menú para registrar un nuevo dispositivo	102
Figura 77. Menú para flashear la imagen de BalenaOS descargada	102
Figura 78. Raspberry Pi 4B con el concentrador RAK2245.....	103
Figura 79. Dashboard que permite gestionar y monitorizar gateway de forma remota.	103
Figura 80. Diseño 3D de la tapa del gateway.....	104
Figura 81. Diseño 3D de la carcasa del gateway.....	105
Figura 82. Parte externa del gateway impreso en 3D	105
Figura 83. Parte interna del gateway impreso en 3D	106
Figura 84. Página principal de TTN	107
Figura 85. Página para registrar dispositivos en la red TTN.....	107
Figura 86. Menú para crear una aplicación en TTN	108
Figura 87. Menú principal para gestionar una aplicación en TTN	108
Figura 88. Menú para registrar un nodo en la red TTN	109
Figura 89. Menú para gestionar el nodo registrado en la red TTN	110
Figura 90. Fragmento del código donde aparece el Device EUI, App EUI y AppKey ...	111
Figura 91. Menú para registrar el gateway en la red TTN	112
Figura 92. Menú para gestionar el gateway registrado en la red TTN.....	113
Figura 93. Interfaz visual que recoge los datos transmitidos por los nodos	114
Figura 94. Conversión de mensajes en formato hexadecimal a formato JSON	115
Figura 95. Página de registro en Datacake.....	117

9. Índice de figuras

Figura 96. Menú de opciones del dashboard Datacake	117
Figura 97. Menú de registro de un nuevo Producto en Datacake.....	118
Figura 98. Menú de registro de un nuevo Producto en Datacake.....	118
Figura 99. Menú para seleccionar el tipo de servidor de red.....	119
Figura 100. Menú para registrar un nuevo dispositivo en Datacake.....	102
Figura 101..Widgets para visualizar los datos.....	120
Figura 102..Histórico de mediciones de CO2 en Datacake.....	120
Figura 103..Histórico de mediciones de la temperatura en Datacake	121
Figura 104.Histórico de mediciones de la humedad en Datacake.....	121
Figura 105.Estructura de nodos y flujos para la gestión de los datos	123
Figura 106. Menú del nodo MQTT TTN.....	124
Figura 107..Menú del nodo change.....	125
Figura 108. Menú del nodo debug.....	126
Figura 109. Menú del nodo mqtt out.....	126
Figura 110. Estructura de nodos y flujos para gestión de alertas y envío mensajes.....	127
Figura 111. Estructura de nodos y flujos para interactuar con el bot de Telegram	129
Figura 112. Pantalla del dispositivo final.....	132
Figura 113. Parte del dashboard que numéricamente los valores medidos.....	134
Figura 114. Parte del dashboard correspondiente a valores medidos.....	135
Figura 115. Parte del dashboard que muestra el valor de la concentración de CO2....	136
Figura 116. Bot de Telegram enviando una alerta.....	136
Figura 117. Bot de Telegram interactuando con el usuario.....	137
Figura 118..Interfaz del Arduino IDE.....	162
Figura 119. Menú para acceder a la opción “Preferencias” del Arduino IDE.....	163

9. Índice de figuras

Figura 120. Gestor de URLs Adicionales de Tarjetas.....	163
Figura 121. Gestor de tarjetas del Arduino IDE.....	164
Figura 122. Menú para elegir la versión de nuestra placa.....	164
Figura 123. Menú para elegir la frecuencia de nuestra placa.....	165

10. Índice de tablas

Tabla 1. Categorías del aire interior en función del tipo de local	14
Tabla 2. Relación del factor de ensanchamiento con otros parámetros	29
Tabla 3. Comparativa entre tecnologías de comunicación LPWA.....	35
Tabla 4. Especificaciones del ESP32-WROOM-32U	38
Tabla 5. Especificaciones del MH-Z19B	44
Tabla 6. Especificaciones del MQ-135.....	46
Tabla 7. Comparativa entre los sensores DHT11 y DHT22	49
Tabla 8. Especificaciones de la Raspberry Pi 4B.....	54
Tabla 9. Comparativa entre gateways LoRaWAN.....	57
Tabla 10. Comparativa entre protocolos IoT	71
Tabla 11. Tabla comparativa de los distintos planes ofertados por Datacake	77
Tabla 12. Comparativa de los distintos planes ofertados por Ubidots.....	80
Tabla 13. Tabla comparativa de los distintos planes ofertados por ThingSpeak	84
Tabla 14. Pines y conexiones del sensor MH-Z19B.....	93
Tabla 15. Pines y conexiones del sensor DHT22	94
Tabla 16. Presupuesto del proyecto	139

11. Referencias

- Actility. (2021, mayo 31). *Indoor Air Quality - why does it matter and how to measure it with IoT?* Actility. <https://www.actility.com/indoor-air-quality-why-does-it-matter-and-how-to-measure-it-with-iot/>
- Adafruit. (2018, marzo). *DHT sensor library*. GitHub. <https://github.com/adafruit/DHT-sensor-library>
- Aernouts, M., Berkvens, R., Van Vlaenderen, K. y Weyn, M. (2018, abril). Sigfox and LoRaWAN Datasets for Fingerprint Localization in Large Urban and Rural Areas. *data*, 3(2), 15. https://www.mdpi.com/2306-5729/3/2/13?type=check_update&version=1
- AlfaIOT. (s.f.). *Arquitectura IoT de 5 capas*. AlfaIOT. <https://alfaiot.com/#:~:text=Arquitectura%20IoT%20de%205%20capas>
- Alfa IOT. (2021, febrero). *Chirpstack vs TTN*. Alfa IOT. <https://alfaiot.com/vs/chirpstack-ttn/>
- Aprendiendo Arduino. (2019, octubre). *Gateways IoT*. Aprendiendo Arduino. <https://aprendiendoarduino.wordpress.com/tag/gateway-iot/>
- Arduino. (2019, diciembre). *SPI library*. Arduino Reference. <https://www.arduino.cc/en/reference/SPI>
- arvindpdmn, M.S, C. y S, A. (2019, septiembre). *Constrained Application Protocol*. Devopedia. <https://devopedia.org/constrained-application-protocol#Dizdarevic-et-al.-2018>
- Asociación Técnica Española de Climatización y Refrigeración. (2012, junio). *Guía técnica de instalaciones de climatización con equipos autónomos*. IDAE.

11. Referencias

- [https://www.idae.es/uploads/documentos/documentos_17_Guia_tecnica_instalacion
es_de_climatizacion_con_equipos_autonomos_f9d4199a.pdf](https://www.idae.es/uploads/documentos/documentos_17_Guia_tecnica_instalacion_es_de_climatizacion_con_equipos_autonomos_f9d4199a.pdf)
- Axiomet. (s.f.). *Monitoreo de la calidad del aire en los espacios cerrados*. Axiomet.
- <https://axiomet.eu/es/es/page/1954/Monitoreo-de-la-calidad-del-aire-en-los-espacios-cerrados/>
- Azzola, F. (2018, noviembre). *CoAP Protocol: Step-by-Step Guide*. DZone Articles.
- <https://dzone.com/articles/coap-protocol-step-by-step-guide>
- BalenaOS. (s.f.). *What is balenaOS?* Balena Reference.
- <https://www.balena.io/docs/reference/OS/overview/2.x/#introduction>
- Bigelow, S. J. (2014, abril). *Ultimate IoT implementation guide for businesses*. IoT Agenda.
- <https://internetofthingsagenda.techtarget.com/Ultimate-IoT-implementation-guide-for-businesses>
- Canvision Systems. (2021, marzo). *Conceptos de actualidad: LoRa y LoRaWan*. 2CI Group.
- <https://www.2cigroup.com/es/conceptos-de-actualidad-lora-y-lorawan/>
- Centro de Enseñanza Academia Testo. (2018). *Absorción infrarroja (Proceso IR)*. Academia testo. <http://www.academiatesto.com.ar/cms/absencion-infrarroja-proceso-ir>
- Chaudhari, B. S., Zenaro, M. y Borkar, S. (2020, marzo 6). LPWAN Technologies: Emerging Application Characteristics, Requirements, and Design Considerations. *Future Internet*, 12(3), 25.
- https://www.researchgate.net/publication/339755491_LPWAN_Technologies_Emerging_Application_Characteristics_Requirements_and_Design_Considerations
- COIIAR. (2021, mayo 24). *GUÍA DE REFERENCIA COVID*. coiilar.
- https://coiilar.es/guiareferenciacovid_coiilar_definitivo/

11. Referencias

- del Valle Hernández, L. (s.f.). *Introducción a Node-RED y Raspberry Pi con un sistema de alarma con Arduino*. Programar Fácil. <https://programarfácil.com/blog/raspberry-pi/introducción-node-red-raspberry-pi/>
- Departamento de Salud Ambiental Subdirección General de Salud Pública. (2020, 10 8). *Medición de la concentración de CO2 como indicador de una ventilación adecuada de edificios y locales. COVID19*. Madrid Salud. https://madridsalud.es/wp-content/uploads/2020/11/InfSAM33-2020Ventilacion_interio_como_medida_preventivaCOVID19.pdf
- Descubre Arduino. (s.f.). *ThingSpeak, plataforma gratuita para la Internet de las Cosas*. Descubreaduino. Retrieved junio, 2021, from <https://descubreaduino.com/thingspeak/>
- Dragino. (2021, enero). *LIG16 LoRaWAN Indoor Gateway*. Dragino Specifications. <https://www.dragino.com/products/lora-lorawan-gateway/item/171-lig16.html>
- Elder, J. (2019, agosto). *How Kevin Ashton named The Internet of Things*. Avast Blog. <https://blog.avast.com/kevin-ashton-named-the-internet-of-things>
- Fernández, Y. (2020, agosto). *Qué es Arduino, cómo funciona y qué puedes hacer con uno*. Xataka. <https://www.xataka.com/basics/que-arduino-como-funciona-que-puedes-hacer-uno>
- Ferrando, M., & Valero, A. (s.f.). Introducción. Parámetros de Antenas. In *ANTENAS* (p. 16). http://www.radiocomunicaciones.net/pdf/introducción_parametros_antenas.pdf
- Foubert, B. y Mitton, N. (2020, enero). Long-Range Wireless Radio Technologies: A Survey. *Future Internet*, 12(1), 17. https://www.mdpi.com/1999-5903/12/1/13?type=check_update&version=2

11. Referencias

García Hernández, S. (2021, febrero 27). *¿Por qué la concentración de CO2 en un lugar*

puede ser indicador de riesgo de contagio de COVID-19? Anadolu Agency.

<https://www.aa.com.tr/es/mundo/-por-qu%C3%A9-la-concentraci%C3%B3n-de-co2-en-un-lugar-puede-ser-indicador-de-riesgo-de-contagio-de-covid-19/2158723>

Guerra Carmenate, J. (2021, marzo). *Cronometrar tiempos con la función millis() y micros()*

de Arduino. Programar Fácil. <https://programarfácil.com/blog/arduino-blog/millis-micros-arduino/>

Hardware Libre. (s.f.). *DHT11: todo sobre el sensor para medir temperatura y humedad.*

Hardware Libre. <https://www.hwlibre.com/dht11/>

Heltec Automation. (s.f.). *Heltec WiFi LoRa 32 (V2) Technical Parameters.* Heltec.

<https://heltec.org/project/wifi-lora-32/>

HelTecAutomation. (2020, mayo). *Heltec_ESP32 Library.* GitHub.

https://github.com/HeiTecAutomation/Heltec_ESP32

Hostingplus. (2020, junio). *Internet de las cosas: qué es y principales características.*

Hostingplus Blog. <https://www.hostingplus.com.es/blog/internet-de-las-cosas-que-es-y-principales-caracteristicas/>

Hübschmann, I. (2020, agosto). *ESP32 for IoT: A Complete Guide.* Nabto.

<https://www.nabto.com/guide-to-iot-esp-32/>

Iberotecno. (2021, febrero). *El sensor de CO2 MQ-135 analizado al detalle.* Iberotecno.com.

<https://iberotecno.com/tienda/blog/electronica/el-sensor-de-co2-mq-135-analizado-al-detalle>

IBM. (2021, mayo). *Qualities of service provided by an MQTT client.* IBM Documentation.

<https://www.ibm.com/docs/en/ibm-mq/8.0?topic=concepts-quality-service>

11. Referencias

IBM. (2021, junio). *Securing AMQP clients*. IBM Documentation.

<https://www.ibm.com/docs/en/ibm-mq/9.2?topic=securing-amqp-clients>

IONOS. (2019, mayo). *AMQP: conoce el Advanced Message Queuing Protocol*. IONOS

Digital Guide. <https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/advanced-message-queuing-protocol-amqp/>

IONOS. (2020, octubre). *10 sistemas operativos para Raspberry Pi*. Digital Guide IONOS.

<https://www.ionos.es/digitalguide/servidores/know-how/sistemas-operativos-para-raspberry-pi/>

Isaac. (s.f.). *MQTT: un protocolo abierto de red y su importancia en el IoT*. Hardware Libre.

https://www.hwlibre.com/mqtt/#Todo_sobre_MQTT

Kerlink. (s.f.). *Kerlink Wirnet iStation*. Kerlink.

https://www.richardsonrfpd.com/docs/rfpd/Wirnet_iStation.pdf

Lanner. (2019, septiembre). *¿Qué es un Gateway IoT?* Lanner Blog. https://www.lanner-america.com/es/blog-es/que-es-un-gateway-iot/#Brindar_seguridad_a_los_datos_del_IoT_anadiendo_una_linea_de_defensa_adicional

https://www.lanner-america.com/es/blog-es/que-es-un-gateway-iot/#Brindar_seguridad_a_los_datos_del_IoT_anadiendo_una_linea_de_defensa_adicional

Lie, R. (Director). (2018). *LoRa/LoRaWAN tutorial 13: Symbol, Spreading Factor and Chip [Film; YouTube]*. Mobilefish.com. <https://www.youtube.com/watch?v=0FCrN-u-Vpw>

The LoRa Alliance. (s.f.). *What are LoRa and LoRaWAN?* <https://lora-developers.semtech.com/library/tech-papers-and-guides/lora-and-lorawan/>

M, A. (2020, abril). *TinkerCAD: ¡Te contamos todo lo que necesitas saber!* 3D natives. <https://www.3dnatives.com/es/tinkercad-software-200420202/#!>

11. Referencias

Massimi, M. (s.f.). *PROTOCOLOS: LA COMUNICACIÓN PARA IOT*. Murky Robot.

<https://www.murkyrobot.com/review/domotica/protocolos-comunicacion-iot>

MCCI Catena. (2021). *Arduino-LMIC library ("MCCI LoRaWAN LMIC Library")*. GitHub.

<https://www.arduino.cc/reference/en/libraries/mcci-lorawan-lmic-library/>

Medina, E. (2019, septiembre). *The Things Network?* The Things Network.

<https://www.thethingsnetwork.org/community/barranquilla/post/the-things-network>

MikroTik. (s.f.). *wAP LR8 kit*. MikroTik Products.

https://mikrotik.com/product/wap_lr8_kit#fndtn-specifications

Mobilefish. (2018, octubre 4). *LoRa/LoRaWAN tutorial 13: Symbol, Spreading Factor and Chip* [Vídeo] [Youtube]. <https://www.youtube.com/watch?v=0FCrN-u-Vpw>

Murata, C., Pant, N. y Iyer, S. (2020, diciembre 18). *Seizing the Trade 4.0 opportunity*.

Deloitte. <https://www2.deloitte.com/us/en/insights/industry/public-sector/trade-4-0-government-opportunity.html>

Nokia Networks. (2015). *[White Paper] LTE-M – Optimizing LTE for the Internet of Things*.

<https://onestore.nokia.com/asset/200178>

Organización Mundial de la Salud (OMS). (2020, noviembre 10). *Información básica sobre la COVID-19. Q&As on COVID-19 and related health topics*.

<https://www.who.int/es/emergencies/diseases/novel-coronavirus-2019/question-and-answers-hub/q-a-detail/coronavirus-disease-covid-19>

Organización Panamericana de la Salud. (2020, marzo 11). *La OMS caracteriza a COVID-19 como una pandemia*. paho. <https://www.paho.org/es/noticias/11-3-2020-oms-caracteriza-covid-19-como-pandemia>

11. Referencias

Organización Panamericana de la Salud. (2020, enero 30). *La OMS declara que el nuevo*

brote de coronavirus es una emergencia de salud pública de importancia

internacional. PAHO Noticias. [https://www.paho.org/es/noticias/30-1-2020-oms-](https://www.paho.org/es/noticias/30-1-2020-oms-declara-que-nuevo-brote-coronavirus-es-emergencia-salud-publica-importancia)

declara-que-nuevo-brote-coronavirus-es-emergencia-salud-publica-importancia

Parada, M. (2017, marzo 24). *Raspberry Pi ya es el ordenador más vendido del mundo.*

Ubuntizando. <https://www.ubuntizando.com/raspberry-pi-ya-es-el-ordenador-mas-vendido-del-mundo/>

Pelaez, A. (2020, diciembre). *5 Reasons Why 2020 Became The Year of IoT.* Ubidots Blog.

<https://ubidots.com/blog/2020-the-year-of-iot/>

Placas de Desarrollo. (s.f.). *Todo sobre las placas de desarrollo para IoT.* Placas de

Desarrollo. <https://www.placasdedesarrollo.com/>

Posey, B., Rosencrance, L., & Shea, S. (2021, marzo). *Industrial Internet of Things (IIoT).*

IoT Agenda. <https://internetofthingsagenda.techtarget.com/definition/Industrial-Internet-of-Things-IIoT>

Pothuganti, K. y Chitneni, A. (2014, Septiembre). A Comparative Study of Wireless Protocols: Bluetooth, UWB, ZigBee, and Wi-Fi. *Advances in Electronic and Electric Engineering (AEEE).*

https://www.researchgate.net/publication/312471356_A_comparative_study_of_wireless_protocols_Bluetooth_UWB_ZigBee_and_Wi-Fi

Radiocrafts. (s.f.). *CoAP - Constrained Application Protocol.* Radiocrafts Technologies.

<https://radiocrafts.com/technologies/coap-constrained-application-protocol/>

11. Referencias

RAK. (2021, mayo). *RAK2245 Pi HAT WisLink LPWAN Concentrator*. RAK Documentation Center. <https://docs.rakwireless.com/Product-Categories/WisLink/RAK2245-Pi-HAT/Overview/#product-features>

Real Academia Española. (s.f.). *Protocolo de comunicaciones*.

<https://dpej.rae.es/lema/protocolo-de-comunicaciones>

Redacción Data Center Market. (2021, febrero). *Sigfox migra su infraestructura a Google Cloud*. Data Center Market.

<https://www.datacentermarket.es/mercado/noticias/1123791032609/sigfox-migra-infraestructura-google-cloud.1.html>

Rivas, J. (2021, junio). *Personalización y Activación en LoRa y LoRaWAN*. LoRa Panamá. <http://lora-panama.com/personalizacion-y-activacion/>

Roland Berger. (2016, mayo). *El reto de la transformación digital de la economía*. Roland Berger.

<https://assets.new.siemens.com/siemens/assets/public.1515407804.4fe796280dd1d58ab6eb71e51f14e13a546c3948.estudio-digitalizacion-espa-a-40.pdf>

Sabas. (2017, octubre). *Haciendo IoT con LoRa: Capítulo 2.- Tipos y Clases de Nodos*. Medium. <https://medium.com/beelan/haciendo-iot-con-lora-capitulo-2-tipos-y-clases-de-nodos-3856aba0e5be>

Sepúlveda, D. (2021, junio). *Integrate your TTN data with Ubidots – Simple Setup*. Ubidots help. <https://help.ubidots.com/en/articles/2362758-integrate-your-ttn-data-with-ubidots-simple-setup>

11. Referencias

- shadowkane. (2019, septiembre). *SX1276 transmission power managment*. The Things Network Forum. <https://www.thethingsnetwork.org/forum/t/sx1276-transmission-power-managment/29720>
- Shelby, Z., Hartke, K. y Bormann, C. (2014, junio). *The Constrained Application Protocol (CoAP)*. Internet Engineering Task Force (IETF). <https://www.rfc-editor.org/rfc/pdfrfc/rfc7252.txt.pdf>
- SigFox. (2017, mayo). *Sigfox Technical Overview*. disk91. <https://www.disk91.com/wp-content/uploads/2017/05/4967675830228422064.pdf>
- Smartbear. (2021, junio). *AMQP Testing*. Smartbear Support. <https://support.smartbear.com/readyapi/docs/testing/amqp.html>
- Stoffregen, P. (2020, abril). *SoftwareSerial*. Platformio Libraries. <https://platformio.org/lib/show/7212/SoftwareSerial/examples>
- Strange_v. (2021, enero). *MHZ19*. GitHub. <https://github.com/strange-v/MHZ19>
- Tezer, O.S. (2013, diciembre). *An Advanced Message Queuing Protocol (AMQP) Walkthrough*. Digital Ocean tutorials. <https://www.digitalocean.com/community/tutorials/an-advanced-message-queuing-protocol-amqp-walkthrough>
- The HiveMQ Team. (2019, agosto). *MQTT Topics & Best Practices - MQTT Essentials: Part 5*. HiveMQ. <https://www.hivemq.com/blog/mqtt-essentials-part-5-mqtt-topics-best-practices/>
- The Things Network. (s.f.). *Building a global open LoRaWAN® network*. The Things Network. Retrieved junio, 2021, from <https://www.thethingsnetwork.org/>

11. Referencias

The Things Network. (2018, febrero). *The Things Network packet forwarder*. GitHub.

https://github.com/TheThingsNetwork/packet_forwarder/tree/legacy#the-things-network-packet-forwarder

ThingSpeak. (s.f.). *ThingSpeak for IoT*. ThinkSpeak pages. Retrieved junio, 2021, from

https://thingspeak.com/pages/commercial_learn_more

T-Mobile. (2019, marzo). [White Paper] *The Game Changer For The Internet Of Things*.

https://www.t-mobile.com/content/dam/tfb/pdf/Whitepaper-Narrow-Bandlot-T2019.pdf?icid=TFB_TMO_P_19IOT_GS082X1D05787BXZN19268

Ubidots. (s.f.). *Ubidots REST API hardware docs*. Ubidots docs. Retrieved junio, 2021, from

<https://ubidots.com/docs/hw/?language=ESP#introduction>

Ubidots. (s.f.). *Ubidots About*. Ubidots. Retrieved junio, 2021, from

<https://ubidots.com/about/>

Unit Electronics. (s.f.). *Especificaciones y Características TTGO LoRa32 V2.1*. Unit

Electronics. <https://uelectronics.com/producto/ttgo-lora32-v2-1-915mhz/>

Universidad de Córdoba. (s.f.). *Tema 4. El efecto invernadero y el calentamiento global*.

<http://www.uco.es/~iq2sagrl/QIMediambiante/TranspTema4-web.pdf>

van Doremalen, N., Morris, D. H. y Holbrook, M. G. (2020, marzo 17). *Aerosol and Surface*

Stability of SARS-CoV-2 as Compared with SARS-CoV-1. The New England Journal of Medicine.

https://www.nejm.org/doi/full/10.1056/nejmc2004973#article_citing_articles

Victor. (2019, julio). *TTN Zaragoza, presentación*. Zaragoza MakerSpace.

<https://zaragozamakerspace.com/index.php/ttn-zaragoza-presentacion/>

11. Referencias

- Visser, H. (2017, abril). *The future of single-channel gateways*. The Things Network Forum. <https://www.thethingsnetwork.org/forum/t/the-future-of-single-channel-gateways/6590>
- Wika. (2020, febrero). *Detector de gas Basado en la tecnología de infrarrojos Modelo GIR-10*. Wika. https://www.wika.es/upload/DS_SP6202_es_es_50945.pdf
- Winsen. (s.f.). *Intelligent Infrared CO₂ Module*. Winsen Sensor. https://www.winsen-sensor.com/d/files/infrared-gas-sensor/mh-z19b-co2-ver1_0.pdf
- Zafra, M., y Salas, J. (2020, octubre 28). *Un salón, un bar y una clase: así contagia el coronavirus en el aire*. elpais. <https://elpais.com/ciencia/2020-10-24/un-salon-un-bar-y-una-clase-asi-contagia-el-coronavirus-en-el-aire.html>

12. Glosario

ABP: Activation By Personalization – Activación Por Personalización

ADC: Analog to Digital Converter – Conversor Analógico Digital

AES: Advanced Encryption Standard – Estándar de Cifrado Avanzado

API: Application Programming Interface – Interfaz de Programación de Aplicaciones

ADR: Adaptive Data Rate - Velocidad de Datos Adaptativa

CPU: Central Process Unit – Unidad Central de Procesamiento

CSS: Chirp Spread Spectrum – Espectro Ensanchado Chirp

dB: Decibelido

dBm: Decibelio milivatio

DBPSK: Differential Binary Phase Shift Keying – Modulación por Desplazamiento de Fase Binaria Diferencial

DSSS: Direct Sequence Spread Spectrum - Espectro Ensanchado de Secuencia Sirecta

eDRX: extended Discontinuous Reception - Recepción Discontinua Ampliada

eMTC: enhanced Machine Type Communications – Comunicaciones de Tipo Máquina Mejoradas

FDD: Frequency Division Duplexing – Duplexación por División de Frecuencia

GFSK: Gaussian Frequency Shift Keying – Modulación por Desplazamiento de Frecuencia Gaussiana

GPS: Global Positioning System – Sistema de Posicionamiento Global

GSM: Global System for Mobile communications – Sistema Global para comunicaciones Móviles

IoT: Internet of Things – Internet de las Cosas

JSON: JavaScript Object Notation – Notación de Objetos JavaScript

kHz: kilohercio

kbps: kilobits per second – kilobits por segundo

LoRa: Long Range – Largo alcance

LoRaWAN: Long Range Wide Area Network - Red de Área Amplia de Largo Alcance

LPWAN: Low Power Wide Area Network - Red de Área Amplia de Bajo Consumo

LTE-M: Long Term Evolution for Machines - Evolución a Largo Plazo para Máquinas

MHz: Megahercio

M2M: Machine to Machine – Máquina a Máquina

NB-IoT: Narrow Band IoT – IoT de Banda Estrecha

NDIR: Non Dispersive Infrared Detector – Detector de Infrarrojos No Dispersion

OTAA: Over The Air Activation – Activación a través del Aire

ppm: partícula por millón

PSM: Power Saving Management - Gestión de Ahorro de Energía

RSSI: Received Signal Strength Indicator

SF: Spreading Factor – Factor de Propagación

SNR: Signal Noise Ratio – Relación Señal a Ruido

SoC: System on a Chip - Sistema en un chip

TCP: Transport Control Protocol – Protocolo de Control de Transmisión

TDD: Time Division Duplexing – Duplexación por División de Tiempo

TTN: The Things Network

UART: Universal Asynchronous Receiver Transmitter – Receptor Transmisor Asíncrono Universal

ANEXO I: Configuración del IDE de Arduino

Para programar el software que permita a la placa de desarrollo recibir los datos de temperatura, humedad y CO₂, se requiere de un entorno de programación (IDE). Hoy en día existen varios IDE que permiten programar placas de desarrollo, como por ejemplo PlatformIO, ESPHome o Arduino IDE. Para programar la placa utilizada en este TFG, se utilizará el Arduino IDE, cuyo lenguaje de programación está basado en C/C++. Este software, además de su sencilla interfaz, destaca por ser compatible con multitud de placas de desarrollo como Arduino, NodeMCU, WEMOS, Heltec ESP32 o CubeCell, entre otras (Fuente: Elaboración propia). Además, es compatible con Windows, Linux y macOS.

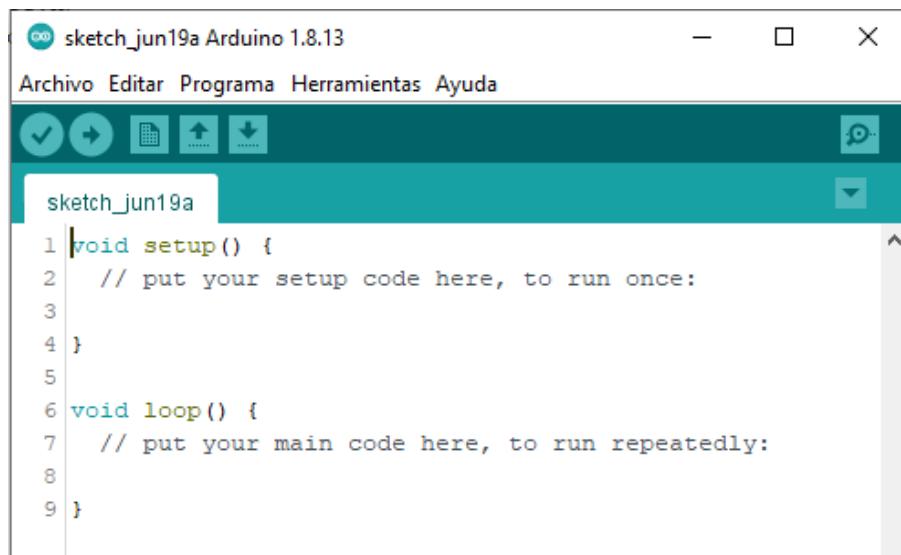


Figura 118. Interfaz del Arduino IDE (Fuente: Elaboración propia)

Para que el IDE de Arduino sea compatible con la placa de desarrollo Heltec Wi-Fi LoRa 32 V2.1, debemos seguir los siguientes pasos:

- 1) Pulsar la pestaña llamada “Archivo”, y a continuación, “Preferencias”.

Sistema IoT de medición de CO₂ para prevención de COVID-19 basado en LoRaWAN

Anexo

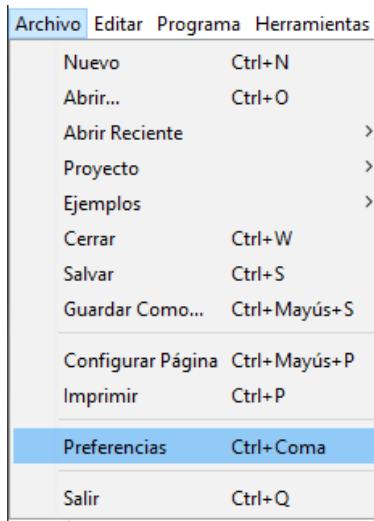


Figura 119. Menú para acceder a la opción “Preferencias” del Arduino IDE (Fuente: Elaboración propia)

- 2) En “Gestor de URLs Adicionales de Tarjetas”, introducimos el siguiente enlace:
https://resource.heltec.cn/download/package_heltec_esp32_index.json.

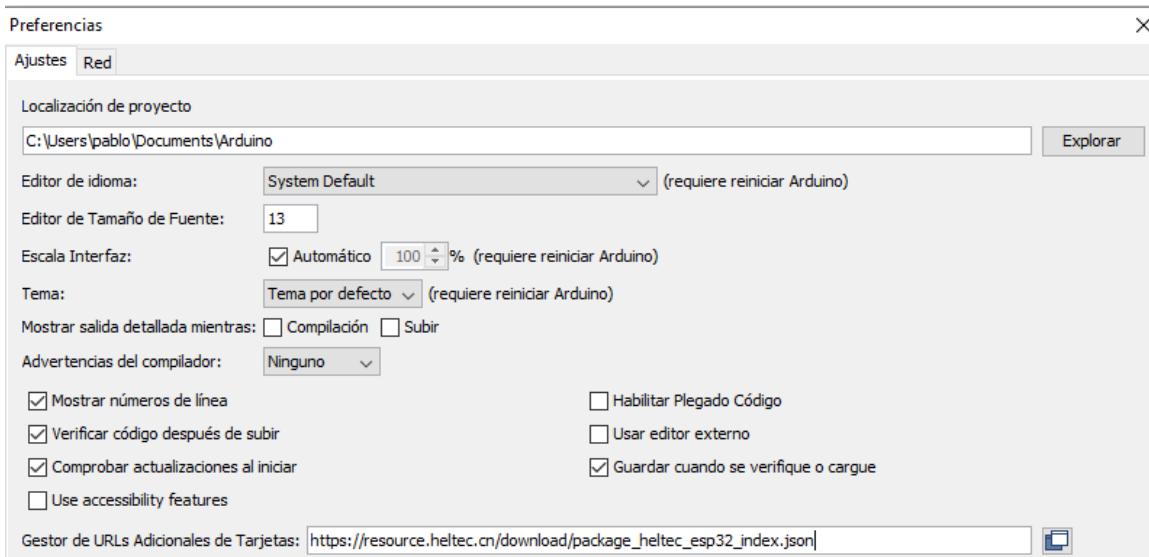


Figura 120. Gestor de URLs Adicionales de Tarjetas (Fuente: Elaboración propia)

- 3) Una vez hayamos introducido el enlace, seleccionamos el botón “Ok” y nos dirigimos a la pestaña “Herramientas”, “Placa”. Desde aquí, seleccionamos la opción de “Gestor de tarjetas” y en la barra de búsqueda escribimos “Heltec ESP32”. Nos mostrará una opción para instalar los archivos necesarios para poder utilizar la placa con el Arduino IDE.



Figura 121. Gestor de tarjetas del Arduino IDE (Fuente: Elaboración propia)

- 4) Cuando se haya completado la instalación, seleccionamos el botón “Cerrar” y a continuación volvemos a pulsar en la pestaña “Herramientas” y “Placa”. Ahora, ya aparecerán las distintas versiones de la placa Heltec ESP32, entre ellas, la versión que se usará en este TFG, Wi-Fi LoRa 32(V2).

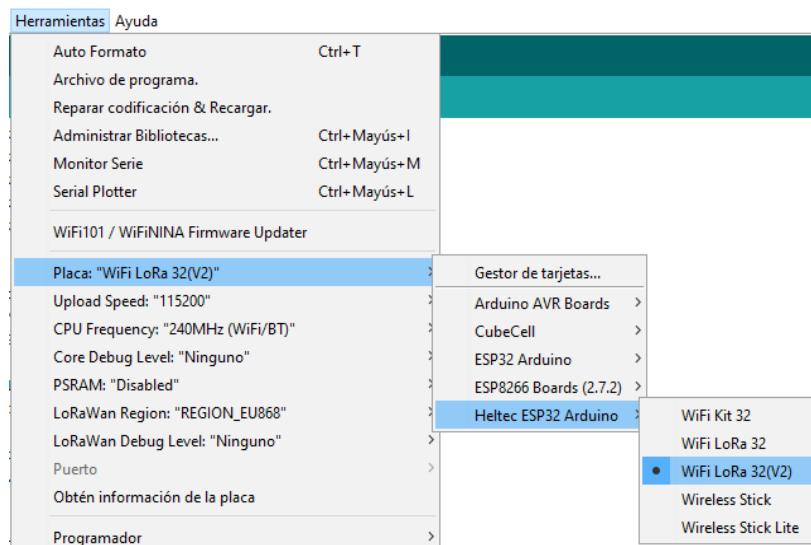


Figura 122. Menú para elegir la versión de nuestra placa (Fuente: Elaboración propia)

- 5) Por último, en la opción “Upload Speed” o velocidad de subida seleccionamos “115200”, siendo este el número de símbolos por segundo, es decir, la velocidad con la que se comunica la placa con el monitor serie del Arduino IDE. Con respecto a la opción “LoRaWan Region”, elegimos “REGION_EU868”, ya que la frecuencia a la que opera LoRa en Europa es 868 MHz. El resto de los valores no se modifican.

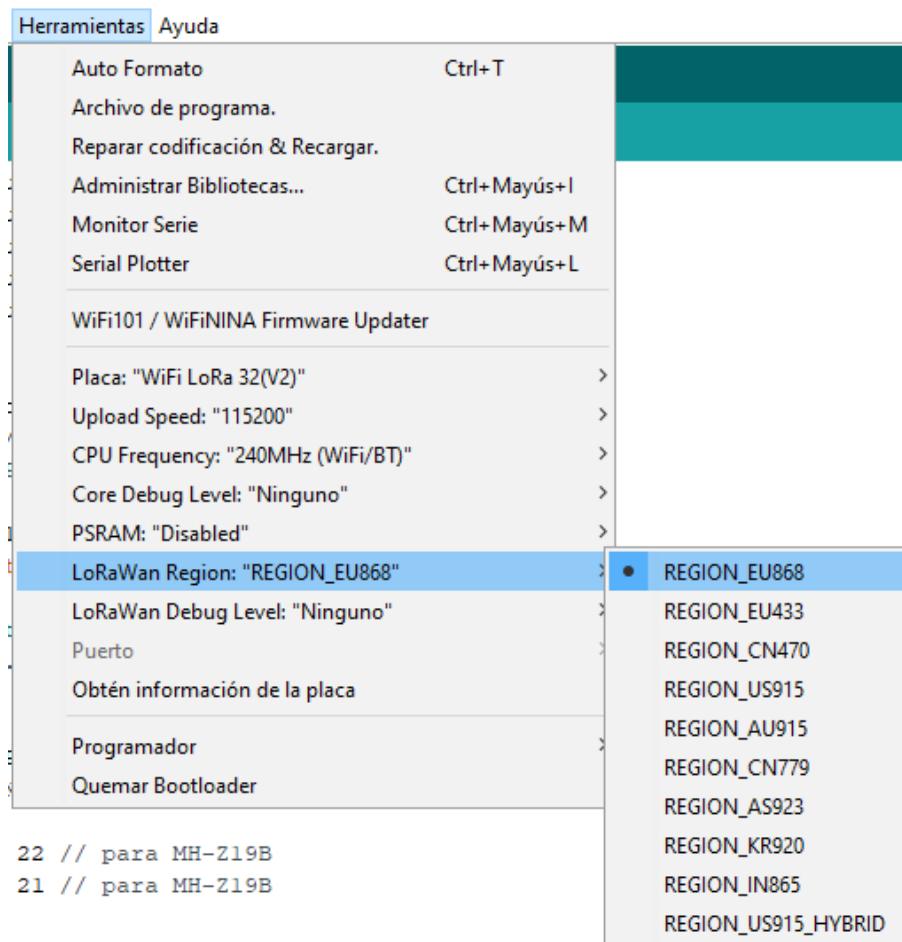


Figura 123. Menú para elegir la frecuencia de nuestra placa (Fuente: Elaboración propia)

Una vez completados todos los pasos descritos anteriormente, el IDE de Arduino ya podría reconocer todos los diferentes modelos de la placa Heltec ESP32.

ANEXO II: Código utilizado

En este apartado se va a describir paso a paso el código que se ha utilizado para programar la placa de desarrollo, cuya descarga está disponible a través del siguiente enlace https://acortar.link/tfg_del_Arco.

En esta primera parte, se han declarado las librerías que nos permiten programar los sensores, entre las que se encuentran:

```
#include <lmic.h>  
  
#include <SPI.h>  
  
#include <MHZ19.h>  
  
#include <SoftwareSerial.h>  
  
#include "heltec.h"  
  
#include "DHT.h"
```

- **lmic.h**: Creada por IBM, la librería LMIC (LoRaWAN-MAC-in-C) permite programar las placas que poseen el chip LoRa SX1272 / 1276 de Semtech para hacerlas compatibles con LoRaWAN y, por ende, con TTN (MCCI Catena, 2021).
- **SPI.h**: Esta librería debe declararse ya que SPI (Serial Peripheral Interface o protocolo de comunicación en serie) es la comunicación que utilizan los microcontroladores para comunicarse con un ordenador o con otro microcontrolador. En este tipo de comunicación síncrona interviene un maestro (normalmente el microcontrolador) y un esclavo (los dispositivos que controla el maestro). Existen 4 tipos de señales (Arduino, 2019):
 - **MISO** (Master In Slave Out): Los datos se transmiten desde el esclavo hasta el maestro.
 - **MOSI** (Master Out Slave In): Los datos se transmiten desde el maestro hasta el esclavo.

- **SCK** (Serial Clock): Este tipo de señal sirve para sincronizar la transmisión de datos procedentes del maestro.
- **MHZ19.h**: Compatible tanto para ESP8266 como ESP32, esta librería contiene el software necesario para configurar el sensor de CO₂ MH-Z19B (Strange_v, 2021).
- **SoftwareSerial.h**: Esta librería permite crear un puerto virtual para la comunicación en serie mediante software, simulando la comunicación mediante hardware UART (Universal Asynchronous Transmitter Receiver). Esta librería es necesaria para que el microcontrolador pueda recibir los datos procedentes del sensor de CO₂ MH-Z19B (Stoffregen, 2020).
- **heltec.h**: Esta librería sirve para programar la pantalla que lleva incorporada la placa de Heltec para mostrar las distintas medidas de CO₂, temperatura y humedad en tiempo real (HelTecAutomation, 2020).
- **DHT.h**: Mediante esta librería podremos obtener los valores de temperatura y humedad registrados por el sensor DHT22 (Adafruit, 2018).

A continuación, se han definido las diferentes variables que se van a utilizar a lo largo del sketch así como los pines de transmisión y recepción del sensor MH-Z19B:

```
#define DHTTYPE DHT22  
  
#define DHTPIN 13  
  
#define RXPin 22 // para MH-Z19B  
  
#define TXPin 21 // para MH-Z19B  
  
unsigned long getDataTimer;  
  
unsigned long counter;  
  
unsigned long paquete;  
  
float h;
```

```
float t;  
float CO2;  
float val_CO2 = 0;
```

Debido a que el sensor MH-Z19B se comunica con la placa mediante la interfaz digital UART, es necesario habilitar un segundo puerto serie ya que el primero de ellos se emplea para que la placa pueda comunicarse con el serial monitor o monitor serie del Arduino IDE. Por tanto, en primer lugar debemos crear un objeto correspondiente a la clase **SoftwareSerial** indicando los pines de transmisión y recepción para poder transmitir los datos por el puerto serie. Posteriormente, creamos una instancia del sensor MH-Z19B mediante un puntero al puerto serie al que está conectado. Con respecto al sensor DHT22, debemos indicar cuál es su pin de datos (PIN 13) así como el modelo, en este caso es el DHT22.

```
SoftwareSerial ss(RXPin, TXPin);  
  
MHZ19 mhz(&ss);  
  
DHT dht(DHTPIN, DHTTYPE);
```

Como el método de autenticación elegido es OTAA, debemos introducir las claves AppEui, DevEui y AppKey. Estas claves se obtienen durante el proceso de registro del dispositivo en la red TTN. En el apartado 5.3.3 se explicará paso a paso este proceso. Tanto la clave AppEui como DevEui deben estar escritas en el formato little-endian (byte menos significativo primero), mientras que la clave AppKey debe estar escrito en big-endian (byte más significativo primero):

```
// lsb (little-endian)  
  
static const u1_t PROGMEM APPEUI[8] = { 0x0E, 0xF8, 0x03, 0xD0, 0x7E,  
0xD5, 0xB3, 0x70 };  
  
void os_getArtEui (u1_t* buf) {
```

```
    memcpy_P(buf, APPEUI, 8);

}

// lsb (little-endian)

static const u1_t PROGMEM DEVEUI[8] = { 0x6F, 0x02, 0x44, 0x08, 0x66,
0xDE, 0xD5, 0x00 };

void os_getDevEui (u1_t* buf) {

    memcpy_P(buf, DEVEUI, 8);

}

// msb (big-endian)

static const u1_t PROGMEM APPKEY[16] = { 0xB2, 0xE2, 0x45, 0x28, 0xE1,
0x5E, 0x38, 0x27, 0x87, 0x65, 0x73, 0x4F, 0xF2, 0xB6, 0x33, 0xB8 };

void os_getDevKey (u1_t* buf) {

    memcpy_P(buf, APPKEY, 16);

}
```

En la variable `payload` debemos indicar cuál es la longitud del mensaje (8 en este caso) que vamos a enviar ya que se trata de un array. `TX_INTERVAL` se refiere al tiempo que transcurre entre un mensaje y el siguiente, en este caso lo hemos programado en 60 segundos:

```
static uint8_t payload[8];

static osjob_t sendjob;

const unsigned TX_INTERVAL = 60;
```

En el siguiente apartado debemos indicar cuáles son los pines que intervienen en la comunicación de LoRa. En nuestra placa, los pines utilizados son el 14, 18, 26, 33, 35.

Sistema IoT de medición de CO₂ para prevención de COVID-19 basado en LoRaWAN

Anexo

```
const lmic_pinmap lmic_pins = {  
    .nss = 18,  
    .rxtx = LMIC_UNUSED_PIN,  
    .rst = 14,  
    .dio = {/*dio0*/ 26, /*dio1*/ 35, /*dio2*/ 33}  
};  
  
void printHex2(unsigned v) {  
    v &= 0xff;  
    if (v < 16)  
        Serial.print('0');  
    Serial.print(v, HEX);  
}
```

La función `onEvent` se ejecuta para notificar ciertos eventos que ocurren durante la conexión entre el nodo y el gateway.

```
void onEvent (ev_t ev) {  
    Serial.print(os_getTime());  
    Serial.print(": ");  
    switch (ev) {  
        case EV_SCAN_TIMEOUT:  
            Serial.println(F("EV_SCAN_TIMEOUT"));  
            break;  
        case EV_BEACON_FOUND:
```

```
Serial.println(F("EV_BEACON_FOUND"));

break;

case EV_BEACON_MISSED:

Serial.println(F("EV_BEACON_MISSED"));

break;

case EV_BEACON_TRACKED:

Serial.println(F("EV_BEACON_TRACKED"));

break;

case EV_JOINING:

Serial.println(F("EV_JOINING"));

break;

case EV_JOINED:

Serial.println(F("EV_JOINED"));

{

u4_t netid = 0;

devaddr_t devaddr = 0;

u1_t nwkKey[16];

u1_t artKey[16];

LMIC_getSessionKeys(&netid, &devaddr, nwkKey, artKey);

Serial.print("netid: ");

Serial.println(netid, DEC);

Serial.print("devaddr: ");

Serial.println(devaddr, HEX);
```

```
Serial.print("AppSKey: ");

for (size_t i = 0; i < sizeof(artKey); ++i) {

    if (i != 0)

        Serial.print("-");

    printHex2(artKey[i]);

}

Serial.println("");

Serial.print("NwkSKey: ");

for (size_t i = 0; i < sizeof(nwkKey); ++i) {

    if (i != 0)

        Serial.print("-");

    printHex2(nwkKey[i]);

}

Serial.println();

}

LMIC_setLinkCheckMode(0);

break;

case EV_JOIN_FAILED:

    Serial.println(F("EV_JOIN_FAILED"));

    break;

case EV_REJOIN_FAILED:

    Serial.println(F("EV_REJOIN_FAILED"));

    break;
```

El siguiente `case` se ejecuta cuando se va a enviar un nuevo mensaje. Concretamente, cuando el método “os_setTimedCallback” llama a la función “do_send”.

```
case EV_TXCOMPLETE:

    Serial.println(F("EV_TXCOMPLETE (includes waiting for RX
windows)"));

    if (LMIC.txrxFlags & TXRX_ACK)

        Serial.println(F("Received ack"));

    if (LMIC.dataLen) {

        Serial.print(F("Received "));

        Serial.print(LMIC.dataLen);

        Serial.println(F(" bytes of payload"));

    }

    os_setTimedCallback(&sendjob, os_getTime() +
sec2osticks(TX_INTERVAL), do_send);

    Serial.println("\n PAQUETE ENVIADO ");

    paquete++;

    break;

case EV_LOST_TSYNC:

    Serial.println(F("EV_LOST_TSYNC"));

    break;

case EV_RESET:

    Serial.println(F("EV_RESET"));

    break;

case EV_RXCOMPLETE:
```

```
    Serial.println(F("EV_RXCOMPLETE"));

    break;

case EV_LINK_DEAD:

    Serial.println(F("EV_LINK_DEAD"));

    break;

case EV_LINK_ALIVE:

    Serial.println(F("EV_LINK_ALIVE"));

    break;

case EV_TXSTART:

    Serial.println(F("EV_TXSTART"));

    break;

case EV_TXCANCELED:

    Serial.println(F("EV_TXCANCELED"));

    break;

case EV_RXSTART:

    break;

case EV_JOIN_TXCOMPLETE:

    Serial.println(F("EV_JOIN_TXCOMPLETE: no JoinAccept"));

    break;

default:

    Serial.print(F("Unknown event: "));

    Serial.println((unsigned) ev);

    break;
```

}

}

La función “do_send” prepara el payload para el momento de la transmisión.

```
void do_send(osjob_t* j) {  
    // Check if there is not a current TX/RX job running  
    if (LMIC.opmode & OP_TXRXPEND) {  
        Serial.println(F("OP_TXRXPEND, not sending"));  
    } else {  
        CO2 = mhz.getCO2();  
        Serial.print("Co2 ");  
        Serial.println(CO2);  
        t = dht.readTemperature();  
        Serial.print("Temperatura: ");  
        Serial.println(t);  
        h = dht.readHumidity();  
        Serial.print("Humedad: ");  
        Serial.println(h);  
        if (isnan(h) || isnan(t)) {  
            Serial.println(F("Error medición DHT22"));  
            return;  
        }  
    }  
}
```

Cuanto mayor sea el tamaño del payload a enviar, mayor consumo de energía y mayor airtime, por lo que la saturación en el canal aumentará y menos mensajes podremos enviar cada hora, como se vió en el apartado 4.1.3. Por tanto, para reducir el tamaño del payload, debemos reducir el tamaño de cada parámetro mediante las siguientes sentencias:

```
uint16_t ft = (uint16_t) (t * 100); // 2 bytes  
uint16_t fh = (uint16_t) (h * 100); // 2 bytes  
uint32_t fco2 = (uint32_t) (CO2 * 100); // 4 bytes
```

Lo que conseguimos es reducir los parámetros de 4 a 2 bytes, como ocurre con la temperatura y la humedad, que son de tipo float. Con respecto al CO₂, como suele variar entre 400 y 5000 ppm, su tamaño se mantiene en 4 bytes. De este modo, el payload tendrá un tamaño de solo 8 bytes:

```
payload[0] = ft >> 8;  
payload[1] = ft % 0xFF;  
payload[2] = fh >> 8;  
payload[3] = fh & 0xFF;  
payload[4] = fco2 & 0xFF;  
payload[5] = (fco2 >> 8) & 0xFF;  
payload[6] = (fco2 >> 16) & 0xFF;  
payload[7] = (fco2 >> 24) & 0xFF;  
LMIC_setTxData2(1, payload, sizeof(payload) - 1, 0);  
Serial.println(F("Packet queued"));  
}}
```

La función “setup” es la primera que se ejecuta cuando arranca el código y solo lo hace una única vez. Dentro de ella se indica la velocidad en baudios de los puertos serie. Como se puede apreciar, la velocidad con la que se comunica la placa con el monitor serie del Arduino IDE es 115200 baudios mientras que la velocidad de transmisión entre el sensor MH-Z19B y la placa es de 9600 baudios.

```
void setup() {  
  
    Serial.begin(115200);  
  
    delay(1000);  
  
    ss.begin(9600);  
  
    Serial.println(F("Starting\n"));
```

Los parámetros incluidos en el método `Heltec.begin()` significan lo siguiente:

- 1º `true`: Inicializa la pantalla que incorpora la placa.
- 2º `true`: Permite que la placa haga uso de su conectividad LoRa para enviar y transmitir datos.
- 3º `true`: Habilita la comunicación en serie.
- 4º `true`: Activa el modo PA Boost que permite aumentar la potencia de transmisión a 20 dBm (shadowkane, 2019).
- 868E6: Hace referencia a la banda de frecuencia de LoRa.

```
    Heltec.begin(true/*DisplayEnable Enable*/, true/*LoRa Enable*/,  
true/*Serial Enable*/, true/*LoRa use PABOOST*/, 868E6/*LoRa RF  
working band*/);  
  
    dht.begin();
```

Este método habilita la autocalibración del sensor MH-Z19B, la cual se detalló en el apartado 4.3.2:

```
mhz.setAutoCalibration(true);
```

El método “os_init()” sirve para inicializar la conexión con la red LoRaWAN, “LMIC_reset()” permite resetear la placa, mientras que “do_send(&sendjob)” ejecuta la función “do_send” e inicia la autenticación OTAA.

```
os_init();  
LMIC_reset();  
do_send(&sendjob);  
}
```

La función “loop” se ejecuta a continuación de la función “setup” y se repite indefinidamente, como un bucle infinito. En su interior, debemos escribir aquellas funciones que nos permitan obtener los distintos parámetros procedentes de los sensores, programar la pantalla de la placa así como declarar una función que compruebe constantemente la comunicación de LoRaWAN.

```
void loop() {
```

El siguiente método comprueba la conexión de LoRaWAN y actualiza su estado continuamente:

```
os_runloop_once();
```

La función `delay()` permite ejecutar una acción cada cierto tiempo. El inconveniente de recurrir a esta función es el hecho de detener completamente la ejecución del código. Esto puede desencadenar diversos problemas ya que hay funciones que comprueban el estado de la

comunicación de LoRaWAN constantemente y si en algún momento estas funciones se detienen, puede haber errores. Para solventar esta problemática, se ha utilizado la función `millis()`, la cual devuelve el tiempo (en milisegundos) que ha transcurrido desde el momento en el que la placa de desarrollo se enciende (Guerra Carmenate, 2021).

Mediante la sentencia `if (millis() - getDataTimer >= 3000)` estamos indicando a la placa que si el tiempo transcurrido desde el arranque es superior o igual a 3000 milisegundos, o lo que es lo mismo, 3 segundos, ejecute el código que hay dentro de la sentencia if:

```
if (millis() - getDataTimer >= 3000) {  
  
    val_CO2 = mhz.getCO2();
```

Si el método `retrieveData()` devuelve el mensaje `MHZ19_RESULT_OK`, significa que el sensor está midiendo correctamente la concentración de CO₂. Si esto ocurre, entonces la sentencia if es cierta y se ejecutará lo que hay en su interior. Todos los parámetros recogidos por los sensores se mostrarán en la pantalla de la placa, gracias a la sentencia `Heltec.display -> drawString()`.

```
MHZ19_RESULT response = mhz.retrieveData();  
  
if (response == MHZ19_RESULT_OK) {  
  
    Heltec.display -> clear();  
  
    Heltec.display -> drawString(0,0,"Co2: "+String(val_CO2)+"ppm");  
  
    Heltec.display -> drawString(0, 10, "Nº de medidas: " +  
String(counter));  
  
    Heltec.display -> drawString(0, 20, "Paquetes enviados: " +  
String(paquete));  
  
    Heltec.display -> display();  
  
}  
  
else {
```

Sistema IoT de medición de CO₂ para prevención de COVID-19 basado en LoRaWAN

Anexo

```
    Heltec.display -> clear();

    Heltec.display -> drawString(0, 0, "ERROR en medición: " +
String(response));

    Heltec.display -> drawString(0, 10, "Nº de medidas: " +
String(counter));

    Heltec.display -> drawString(0, 20, "Paquetes enviados: " +
String(paquete));

    Heltec.display -> display();

}

counter++;

recogerTemp();

getDataTimer = millis();

}

}

La función recogerTemp() permite leer los datos de temperatura y humedad y mostrarlos en la pantalla de la placa.
```

```
void recogerTemp() {

    h = dht.readHumidity();

    t = dht.readTemperature();

    if (isnan(h) || isnan(t)) {

        Serial.println(F("Failed to read from DHT sensor!"));

        return;

}
```

Sistema IoT de medición de CO₂ para prevención de COVID-19 basado en LoRaWAN

Anexo

```
Heltec.display -> drawString(0,30,"Temperatura: "+String(t)+ " °C");  
Heltec.display -> drawString(0, 40, "Humedad: " + String(h) + " %");  
Heltec.display -> display();
```