escolauniversitaria
POLITÉCNICA

UNIVERSIDADE DA CORUÑA

TÍTULO DISEÑO DE SISTEMA DE MONITORIZACIÓN DE CALIDAD DEL AIRE EN LOCALES CON VENTILACIÓN LIMITADA.

MÁSTER MÁSTER EN INFORMÁTICA INDUSTRIAL Y ROBÓTICA

ASIGNATURA TRABAJO FIN DE MÁSTER (Nº 4538M01A001)

ALUMNO DAVID ADRIÁN RODRÍGUEZ GARCÍA

TUTORES FRANCISCO ZAYAS GATO

HÉCTOR QUINTIÁN PARDO

FECHA JUNIO DE 2021

Índice

Listado de figuras

Listado de tablas

Resumen

El presente Trabajo Fin de Máster tiene por objetivo el diseño de un sistema de monitorización de calidad del aire, a partir de hardware y sensórica de bajo coste. Se propone el uso de la plataforma Arduino o similar junto con sensores de CO2, así como de algún sistema de centralización local y/o en la nube para almacenar y visualizar la información capturada.

Resumo

O presente Traballo Final de Máster ten por obxecto o deseño dun sistema de monitorización da calidade do aire, a partir de hardware e sensórica de baixo custo. Proponse o uso da plataforma Arduino ou similar xunto con sensores de CO2, así coma algún sistema de centralización local e/ou na nube para o almacenamento e visualización da información recollida.

Resumo

O objetivo desta Trabalho Final de Mestrado é projetar um sistema monitoramento da qualidade do ar, com base em hardware e sensor de baixa custo. O uso da plataforma Arduino ou similar é proposto em conjunto com sensores de CO2, bem como algum sistema de centralização local e / ou na nuvem para armazenar e ver as informações capturadas.

Abstract

The objective of this Master's Thesis is to design an air quality monitoring system, based on low cost hardware and sensors. Using the Arduino platform or similar is proposed together with CO2 sensors, as well as some local and / or cloud centralization system to store and view the information captured.

1 Introducción

1.1 Introducción

El contexto socio-económico mundial ha cambiado recientemente a causa de la pandemia provocada por el virus *SARS-CoV-2* cuyo modo de infección y propagación es de tipo respiratorio. En esta situación, el control de la calidad del aire que respiramos en recintos cerrados ha cobrado especial importancia para evitar la transmisión del virus y asegurar el correcto desarrollo de la actividad social y empresarial.

Este Trabajo Fin de Máster pretende aportar una solución tecnológica y asequible enmarcada dentro de la rama de desarrollo impulsada por los diferentes Estados para afrontar la recuperación, la **Digitalización e Industria 4.0**.

La propuesta tecnológica a desarrollar en esta memoria trata sobre el diseño de un sistema inteligente de monitorización de la calidad del aire bajo los principios del código libre y el hardware libre y de bajo coste. Buscando con ellos el acceso a esta tecnología a escuelas, universidades, pequeñas y medianas empresas, y demás espacios interiores que necesiten controlar la calidad de su aire interior para con ello prevenir y mejorar la ventilación de estos espacios y asegurar el correcto estado de salubridad de sus instalaciones y de las personas que las frecuentan.

El desarrollo del proyecto se ha publicado de forma libre bajo licencia *GNU General Public License v3.0* que permite su modificación y difusión respetando su continuación como código abierto al público.

Puede accederse al repositorio público a través de la siguiente referencia [8].

1.2 Objetivos

Los objetivos que se plantean para este Trabajo Fin de Máster consisten en el desarrollo de un sistema de monitorización de la calidad del aire mediante la utilización de hardware y sensores de bajo coste así como haciendo uso de servicios de software libre para el tratamiento, visualización y almacenamiento de los datos.

De esta forma, se desarrollará una solución económica que permitirá a diferentes escuelas, oficinas y otros recintos cerrados controlar en todo momento la concentración de CO2 en el aire, con el objetivo de mejorar la ventilación de dichos recintos y reducir con ello las probabilidades de transmisión de enfermedades respiratorias y el agotamiento que produce una insuficiente calidad en el aire que respiran las personas.

1.3 Alcance

El alcance de este Trabajo Fin de Máster comprende el diseño de un sistema de monitorización de la calidad de aire basado las siguientes tecnologías y procesos:

- Análisis de los parámetros del aire a monitorizar.
- Estudio de mercado y selección del mejor sensor calidad-precio.

- Programación y montaje del prototipo de sensor de CO2 inteligente mediante hardware de bajo coste e impresión 3D.
- Despliegue y configuración de los servicios backend en la nube para la recogida, análisis y tratamiento de los datos enviados por el sensor.
- Análisis de los resultados obtenidos y propuestas de mejora del producto de cara a una posible versión comercial del mismo.

1.4 Antecedentes

El sensor de CO2 inteligente de este proyecto aglutina el uso de diferentes tecnologías en un mismo producto, en concreto combina el uso de la plataforma Arduino para programación y como estandarte del hardware de bajo coste, el empleo de comunicaciones IoT que permiten interconectar gran cantidad de objetos cotidianos a internet y la computación en la nube como solución que permite acceder a las ventajas del uso de servidores sin disponer de infraestructura propia, reduciendo notablemente los costes, limitándolos al pago por uso de los recursos.

1.4.1 Plataforma Arduino

Arduino es una plataforma de hardware-libre y código abierto utilizada para construir proyectos de electrónica y para el prototipado rápido de productos. La plataforma cuenta con una gran comunidad de usuarios que respaldan el proyecto y lo alimentan con una amplia variedad de librerías y ports de otros microcontroladores a esta plataforma, como el caso de los microcontroladores para el IoT de Espressif Systems.

La empresa Arduino aporta valor al mercado en dos aspectos, en primer lugar, el desarrollo de placas PCB que contienen todo lo necesario para desarrollar prototipos de electrónica a un coste muy bajo, la más popular es el modelo **Arduino UNO** lanzado inicialmente en su primera versión en el año 2010, aunque el primer modelo de la compañía el Arduino Serie fue lanzado en 2005.

En segundo lugar, la compañía se encarga del mantenimiento y desarrollo del popular IDE de Arduino que permite la programación de todas su placas y las de otros fabricantes que hayan portado sus librerías a este entorno de desarrollo. El lenguaje de programación utilizado es una variante de C/C++ que solo requiere un mínimo de dos funciones básicas para funcionar, la función `setup` que se ejecuta una sola vez al encenderse la placa y la función `loop` que se repite constantemente mientras la placa esté encendida.

Esta plataforma ha servido y sirve como base para muchos de los proyectos de electrónica que han surgido en los últimos 10 años, incluido el descrito en esta memoria.

1.4.2 Internet de las Cosas

El concepto de **Internet de las Cosas** o en inglés **Internet of Things (IoT)** surgió en el año 1999 en el MIT en el campo de la identificación por radiofrecuencia (RFID), pero hoy en día es un término mucho más amplio que comprende gran cantidad de objetos que pueden ampliar sus funcionalidades mediante una conexión a la red. En este sentido han salido al mercado todo tipo de dispositivos como lavadoras, neveras, bombillas, termostatos, etc. denominados inteligentes que permiten su telegestión desde cualquier lugar a través de internet.

El internet de las cosas presenta aplicación en multitud de campos diferenciados, desde controlar mediante sensores el campo en la agricultura para conseguir las condiciones idóneas de humedad, luz, temperatura, etc. hasta sistemas más complejos como el Automóvil Conectado que cuenta con un gran número de sensores que envían información en tiempo real sobre el estado del vehículo (aceleración, velocidad, nivel de combustible, ubicación, etc.).

En un mundo cada vez más digital el auge del internet de las cosas continúa su crecimiento de forma exponencial, cada días más dispositivos se conectan a la red por lo que la tecnología debe seguir avanzando para garantizar la seguridad de las comunicaciones y la mejora de las mismas, mediante nuevos protocolos y nuevos sistemas de telecomunicaciones como el 5G.

1.4.3 Computación en la nube

Se entiende por **computación en la nube** como un paradigma en el que se ofrecen servicios de cómputo informáticos mediante un modelo de negocio en el que se paga por uso, se solicita el uso de recursos informáticos, principalmente almacenamiento de datos y capacidad de cómputo en un centro de datos que se encuentran disponibles a través de internet.

Dentro de este amplio concepto, existen otros tres que se han vuelto muy populares en los últimos años:

- **Infraestructura como servicio (IaaS)**, es el modelo de pago por uso de servicios informáticos de procesamiento, almacenamiento y redes en los servidores de centros de datos del proveedor del servicio.
- **Plataforma como servicio (PaaS)**, es similar al IaaS pero con una capa superior de abstracción, además de la infraestructura, el servicio contratado ofrece otras herramientas como middlewares, sistemas de Business Intelligence, sistemas de administración de bases de datos y el sistema operativo y diferentes aplicaciones o librerías que son necesarias para el despliegue de la aplicación.
- **Software como servicio (SaaS)**, es un modelo de distribución de software en el que el código y los datos se almacenan en un servidor en la nube y el usuario accede al servicio a través de internet. Normalmente el software puede ser consultado desde cualquier dispositivo a través de un navegador web.

Las aplicaciones más modernas comienzan a ser desarrolladas entorno a estos paradigmas mencionados, y los modelos de negocio cambian de la venta de licencias tradicional a las suscripciones por uso del servicio (SaaS). La figura 1.1 muestra las diferencias entre los 3 conceptos anteriores.

1.4.4 Medidores de CO2 de referencia

Se ha tenido en cuenta como referencia para la realización de este proyecto, algunos de los medidores de CO2 que se venden actualmente en el mercado.

Uno de ellos ha sido el modelo PCE-CMM 5 [9] de la empresa PCE Instruments, se trata de un modelo de medidor de CO2 con pantalla LCD y tecnología NDIR, muestra los valores de concentración de CO2, temperatura y humedad por dicha pantalla pero no dispone de comunicación con internet a diferencia del prototipo a diseñar en esta memoria.

Figura 1.1: Modelos de negocio de computación en la nube

En cuanto a medidores de referencia conectados a internet, existen varios proyectos que circulan por la red a cerca de medidores de CO2 con conexión por internet, entre ellos se ha tomado como referencia especialmente el proyecto **CODOS** [12] de Miguel Ángel Casanova que da una solución similar a la aquí planteada utilizando hardware de bajo coste y un portal cautivo para la visualización de los valores medidos, aunque sin utilizar los servicios cloud que se comentarán en el apartado 4.2.1 y que será la mayor innovación del proyecto descrito en esta memoria.

1.5 Estructura del TFM

La estructura se divide en capítulos para tratar de alcanzar en ellos los objetivos descritos previamente.

En el capítulo 2 se presentan los diferentes motivos que impulsan este Trabajo Fin de Máster en relación con la monitorización de la calidad del aire y los beneficios que ésta puede tener sobre la salud de las personas, además se comentarán los principales parámetros a controlar y se realizará un análisis del mercado de sensores de CO2 y la selección del más apropiado para este proyecto.

En el capítulo 3 se describe el microcontrolador utilizado para la programación del sensor inteligente y los motivos de su elección, además se desarrollan las explicaciones y esquemas necesarios para el conexionado de los diferentes componentes que conformarán dicho sensor inteligente. También se explicará el funcionamiento del programa desarrollado para dicho microcontrolador y el tipo de comunicaciones IoT utilizadas.

En el capítulo 4 se aborda la arquitectura del servidor en la nube necesario para la recogida, almacenamiento y visualización de los datos enviados por el sensor, se explicarán los diferentes servicios y métodos de despliegue utilizados.

En el capítulo 5 se comentan los resultados obtenidos a lo largo de todo el proyecto realizado, se proponen una serie de mejoras sobre el prototipo que podrían incluirse en un producto comercial final y se resumen las conclusiones más importantes obtenidas de este proyecto.

2 Monitorización de la calidad del aire

2.1 Beneficios del control de la calidad del aire

La monitorización y control de la calidad de aire en locales cerrados presenta diferentes beneficios para la salud de las personas, en particular se tratará dicha monitorización en aulas de colegios, institutos o universidades, pues se busca la utilidad del sistema proyectado como medida de monitorización del aire y prevención de enfermedades respiratorias como el *COVID-19* en las instalaciones de la *Universidade da Coruña*.

La información recogida en esta sección ha sido obtenida del documento oficial elaborado por el *CSIC-IDAEA, Ministerio de Ciencia e Innovación y Mesura* [11] y plantea estrategias para indicar las condiciones de ventilación adecuadas en las aulas, así como una serie de características deseables en los medidores de CO2 a utilizar para la monitorización de la calidad del aire.

La concentración de CO2 en espacios interiores cerrados aumenta rápidamente en presencia de personas, que exhalan al respirar. La renovación del aire con aire exterior reduce las concentraciones de CO2 en el interior. Esto favorece la atención y rendimiento escolar, ya que la exposición a concentraciones de CO2 demasiado elevadas produce aletargamiento y dificulta la atención.

Por ello es necesario garantizar la correcta renovación del aire del aula para disminuir la concentración de CO2 y otras partículas infecciosas como los virus. Para medir la renovación de aire en un espacio cerrado se utiliza el término ACH *Air Changes per Hour* que indica que en 1 hora entra en la sala una cantidad de aire igual al volumen de la misma.

Una medida recomendable para el ACH es de 14 litros por persona y segundo que se puede transformar a renovaciones por hora mediante la siguiente expresión.

$$ACH = \frac{litros}{persona \cdot segundo} \cdot n_{personas} \cdot 3600 \frac{seg}{h} \cdot 0.001 \frac{m^3}{litros \cdot V_{sala}}$$

Teniendo en cuenta que la concentración de CO2 es aproximadamente de 420 ppm en ambientes exteriores, y que se entiende como de cierto riesgo sobrepasar los valores de 1000 ppm en un espacio cerrado, la figura 2.1 muestra un ejemplo de la variación de CO2 en un aula con 21 alumnos y un método de ventilación natural con ventilación cruzada (apertura de puertas y ventanas opuestas) que es la que se presenta en la mayoría de las aulas.

Como conclusión obtenida de la gráfica vemos que utilizando adecuadamente la ventilación en el momento oportuno se puede evitar sobrepasar el valor de riesgo de 1000 ppm de concentración de CO2. Teniendo esto en cuenta, el objetivo de este Trabajo de Fin de Máster será desarrollar un sensor de CO2 inteligente que notifique a los usuarios de la sala mediante gráficas en tiempo real y un chat de mensajería, de esta forma podrá preveerse el aumento de la concentración de CO2 y actuar con anterioridad para estabilizar dicha concentración en valores saludables para las personas.

2.2 Principales parámetros a controlar

Como se ha comentado en el apartado 2.1 la medición de la concentración de CO2 es el principal parámetro a controlar en un sistema de monitorización de la calidad de aire, pues permite calcular a partir de ella las renovaciones hora del espacio.

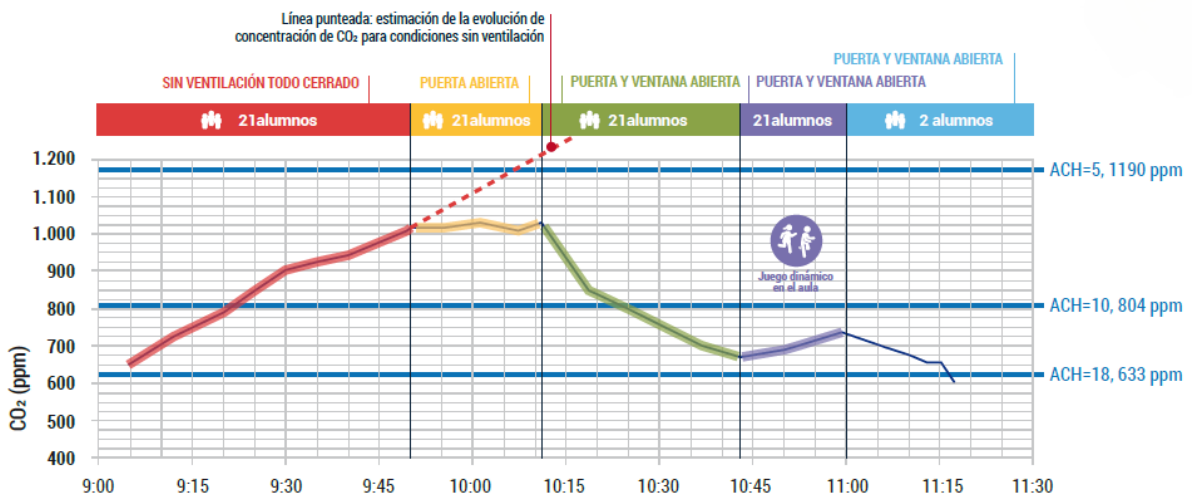

Figura 2.1: Evolución de la concentración de CO2 en un aula

Como parámetros adicionales interesantes que afectan al confort en un espacio interior se podría realizar la medición de las condiciones de temperatura y humedad del ambiente, y de esta forma tratar de garantizar el compromiso ideal entre la ventilación realizada y el confort térmico para las personas, evitando, por ejemplo, una ventilación natural excesiva que provoque una fuerte caída en la temperatura de la sala.

También es conveniente destacar que el CO2 no se degrada con el tiempo, mientras que los virus en aire sí, por lo que las concentraciones de virus en aire decrecerán más rápidamente que las de CO2. La diferencia depende de varios factores ambientales, tales como radiación UV o temperatura. Por lo que utilizar la concentración de CO2 como medida para cuantificar las necesidades de ventilación de un espacio es conservador y permite garantizar la correcta calidad del aire en la sala.

Con referencia a lo aquí mencionado se estudiará el mercado de sensores de CO2 que existe en la actualidad y se tratará de seleccionar el que mejor se adapte en relación calidad-precio a las características mencionadas en el apartado 2.3.

2.3 Sensores de medición de CO2

En este apartado se realizará un estudio de mercado acerca de los diferentes sensores de CO2 existentes para desarrollo/prototipado analizando sus características tanto para el tipo de medición de CO2 que realizan como sus interfaces de comunicaciones con el microcontrolador y el rango de precio en el que se sitúan. Estas características serán comparadas con las recomendadas en el artículo [11] para medidores de CO2 en aulas y son las siguientes:

- Capacidad de proporcionar los datos sin procesar descargables en archivo .txt, .xls, .csv o similar.
- Resolución temporal de al menos un dato por minuto
- Pantalla que muestre los niveles de CO2 en tiempo real
- Uso de tecnología NDIR (del inglés nondispersive infrared)
- Coste entre 100 y 300 €

Estas características antes mencionadas hacen referencia a un medidor de CO2 comercial, el objetivo de este proyecto es aportar una solución diferente y más innovadora, añadiendo conectividad al sensor de CO2 y un conjunto de servicios en la nube para el procesado, almacenamiento y visualización en tiempo real de los valores de concentración medidos. Por lo que aunque han de tenerse en cuenta las características mencionadas, no se seguirán al pie de la letra a la hora de seleccionar el sensor a utilizar.

Del estudio de mercado realizado se han seleccionado los sensores de CO2 más representativos, listando en la tabla 2.1 sus diferentes características técnicas para su comparación.

Existen principalmente 2 tipos de tecnologías que permiten medir la concentración de CO2 en al aire ambiente, se trata de las tecnologías **NDIR (Non Dispersive Infrared)** y **MOS (Metal Oxide Semiconductor)** que se describirán a continuación.

La tecnología NDIR realiza la medición directa del CO2 mediante un haz de luz infrarroja en el interior de la cámara de aire del sensor, el CO2 absorbe una determinada longitud de onda del infrarrojo y de esta forma el detector de infrarrojos con el que cuenta el sensor es capaz de filtrar el resto de longitudes de onda de la luz y medir la intensidad de la longitud de onda que absorbe la molécula de CO2 para con ello obtener su concentración. La figura 2.2 muestra el proceso descrito.

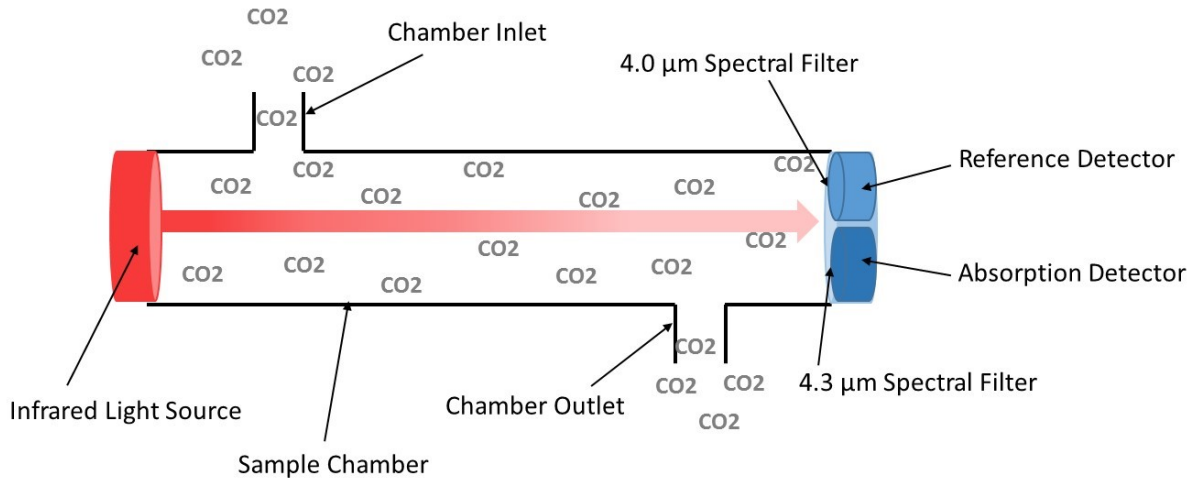

Figura 2.2: Tecnología NDIR en un sensor de CO2

En cambio la tecnología MOS se basa en la medición de los compuestos orgánico volátiles (VOC) del aire por oxidación de un semiconductor de óxido de metal, y con ellos pueden calcular de forma indirecta la concentración de CO2 en el aire, por lo que su precisión es más baja, y depende mucho de los valores de temperatura y humedad a la hora de realizar dicho cálculo lo que implica que deben acompañarse de un sensor externo de Tª y Humedad que suministre dichos valores al sensor MOS.

Las ventajas de los sensores MOS es que suelen ser más baratos que los NDIR, de menor tamaño y menor consumo. En cambio los sensores NDIR ofrecen una medida del CO2 más precisa aunque también su consumo, coste y tamaño es mayor.

Dado que entre las características que se tendrán en cuenta para este proyecto el consumo no es relevante pues el microcontrolador WiFi que se utilizará para comunicar los datos tiene un elevado consumo debido al protocolo **WiFi**, se optará entonces por diseñar un medidor de CO2 inteligente con alimentación directa de la red. En el caso de optar por soluciones inalámbricas con batería sería necesario utilizar otro protocolo de menor consumo como **Zigbee** o **BLE (Bluetooth Low Energy).**

El tamaño, aunque importante para un futuro producto final, no es demasiado relevante en este prototipo y tampoco sería crítico en un producto final, ya que se busca un producto que se ubicará de forma estática en una sala y no es necesaria demasiada portabilidad del mismo.

El coste sí es importante pues en general para la monitorización de la calidad de aire de una escuela o universidad sería necesario contar con al menos uno por aula, por lo que reducir al máximo los costes supondría una mayor posibilidad de aceptación del producto hacia este segmento de clientes objetivo.

La precisión en la medición juega un papel clave a la hora de seleccionar el tipo de tecnología del sensor, lo que se busca es la previsión de la concentración de CO2 con anterioridad y un correcto control de su evolución en función de la ventilación realizada por lo que es necesaria una buena precisión que garantice una correcta correlación entre la concentración y el uso de la ventilación.

Teniendo en cuenta todas estas características descritas y el estudio de mercado realizado y recogido en la tabla 2.1 se ha decidido seleccionar para este proyecto el modelo de sensor de tecnología NDIR **MH-Z19c** de la marca Winsen pues permite asegurar una correcta relación calidad-precio.

Fabricante	Modelo	Tecnología	Rango Efectivo	Precisión	Interfaz	Tiempo de respuesta	Sensores	Precio
Amphenol	T6713	NDIR	400-2000 ppm	± (25 ppm + 3 %)	UART, I2C, PWM	<180 s (90 %)	CO2	48.19 - 70.99 €
Amphenol	T6713-5K	NDIR	400-5000 ppm	± (30 ppm + 3 %)	UART, I2C, PWM	<180 s (90 %)	CO2	51.46 - 70.99 €
Cubic	CM1106-C	NDIR	400-5000 ppm	± (50 ppm + 5 %)	UART, I2C, PWM	30 s	CO2	
Cubic	CM1106SL-NS	NDIR	400-5000 ppm	± (50 ppm + 5 %)	UART	120 s	CO2	
Gas Sensing Sol.	COZIR-AH-E-1	NDIR	400-10000 ppm	± (30 ppm + 3 %)	UART	30 s (90 %)	CO2, H, T	73.78 - 100.14 €
Gas Sensing Sol.	COZIR-LP-5000	NDIR	400-5000 ppm	± (30 ppm + 3 %)	UART	30 s (90 %)	CO2	63.41 - 86.06 €
Honeywell	CRIR M1	NDIR	400-2000 ppm	± (40 ppm + 3 %)	UART	120 s	CO2	39.48 - 51.34 €
SenseAir	S8 LP	NDIR	400-2000 ppm	± (40 ppm + 3 %)	UART, PWM	<120 s (90 %)	CO2	23.17 - 36.42 €
SenseAir	Sunrise	NDIR	400-5000 ppm	± (30 ppm + 3 %)	UART, I2C	16 s	CO2	44.84 €
Sensirion	SCD-41	NDIR	400-5000 ppm	± (40 ppm + 5 %)	I2C	60 s (63 %)	CO2, H, T	42.94 €
Sensirion	SCD-30	NDIR	400-10000 ppm	± (30 ppm + 3 %)	UART, I2C, PWM	20 s (63 %)	CO2, H, T	30.57 - 44.47 €
Winsen	MH-Z19B	NDIR	400-5000 ppm	± (50 ppm + 5 %)	UART, PWM	<120 s (90 %)	CO2	18.61 €
Winsen	MH-Z19C	NDIR	400-5000 ppm	± (50 ppm + 5 %)	UART, PWM	<120 s (90 %)	CO2	17.09 €
ZyAura (Radiant)	ZG09	NDIR	400-10000 ppm	± (50 ppm + 3 %)	UART, I2C, PWM	60s	CO2	27.11 €
Adafruit	AMS CCS811	MOS	400-8192 ppm	± 2 %	I2C	250ms - 60s	CO2	24 €
Sparkfun	SEN-16531	MOS	400-60000 ppm	± 1 %	I2C	1s	CO2	20 €
Keystudio	AMS CCS811	MOS	400-8192 ppm	± 2 %	I2C	250ms - 60s	CO2	15 €
Sparkfun	SEN-14348	MOS	400-8192 ppm	± 2 %	I2C	250ms - 60s	CO2, H, T	29 €

Tabla 2.1: Comparativa de sensores comerciales de CO2

2.3.1 Sensor MH-Z19C

El sensor de la empresa China **Zhengzhou Winsen Electronics Technology Co., Ltd** en concreto el modelo **MH-Z19c** es el que se ha seleccionado dada su alta relación calidad-precio.

Se trata de un sensor que utiliza la tecnología NDIR, con un rango efectivo entre 400 ppm y 5000 ppm, que cubre perfectamente la aplicación para la que está destinado, lo mismo sucede con la precisión ± (50 ppm + 5 %), la interfaz de comunicación es UART y cuenta con multitud de librerías que facilitan el desarrollo de código para el entorno Arduino, como se explicará en la sección 3.3.3, además también presenta la interfaz de comunicación PWM aunque no se usará en este proyecto.

Como puntos más negativos del sensor es que sólo cuenta con sensor de CO2, no incluye de humedad y temperatura, pero puede calcular la temperatura de forma indirecta en función de la concentración de CO2 medida. Además el tiempo de respuesta del sensor es de 120 segundos, el doble de lo recomendado en 2.3, éste será el parámetro limitante a la hora de seleccionar el tiempo de muestreo en el programa del microcontrolador.

En general sopesando todas las características y teniendo muy en cuenta su principal ventaja, el precio, es el sensor NDIR más barato del mercado, apenas unos 17,09 €, es el sensor ideal para un producto que pretende dar una solución económica, innovadora y precisa para el control de la calidad de aire en las aulas de escuelas y universidades.

Una documentación más extensa acerca de las características técnicas del sensor puede encontrarse en la ficha técnica del fabricante [6].

3 Electrónica y comunicaciones IoT

3.1 Placa de desarrollo NodeMCU V3

La placa electrónica PCB encargada de hacer “inteligente” el sensor de CO2 seleccionado en el apartado 2.3.1, se trata de una placa **NodeMCU V3** que es una plataforma *IoT* de código abierto y hardware libre que incluye el SoC ESP8266 de la empresa **Espressif Systems**.

NodeMCU surgió inicialmente como un kit de desarrollo para el prototipado de dispositivos *IoT* mediante el lenguaje de scripting **LUA**, posteriormente al desarrollo de la portabilidad del core de Arduino para ESP8266, NodeMCU amplió su compatibilidad con éste entorno de desarrollo, que es el que utilizaremos para este proyecto como se describe en la sección 3.3.

La placa cuenta con el SoC ESP8266 como chip principal, con sus correspondientes GPIOs mapeados a las entradas/salidas digitales y la entrada analógica de la placa que se mencionarán más en detalle en el apartado 3.1.1, cuenta también con una interfaz micro USB para su programación y alimentación, además del regulador de tensión que permite alimentarlo a 5V desde el micro USB o hasta 12V desde el pin `Vin`. Cuenta también con un pin de salida `VU` conectado directamente a la tensión del micro USB por lo que podría resultar muy útil para alimentar directamente sensores o actuadores que funcionen a 5V en lugar de los 3.3V de operación del SoC.

Como conversor de USB a interfaz serie para programar el SoC la placa incluye el chip FTDI modelo `CH340`, la siguiente figura 3.1 muestra el pinout de la placa.

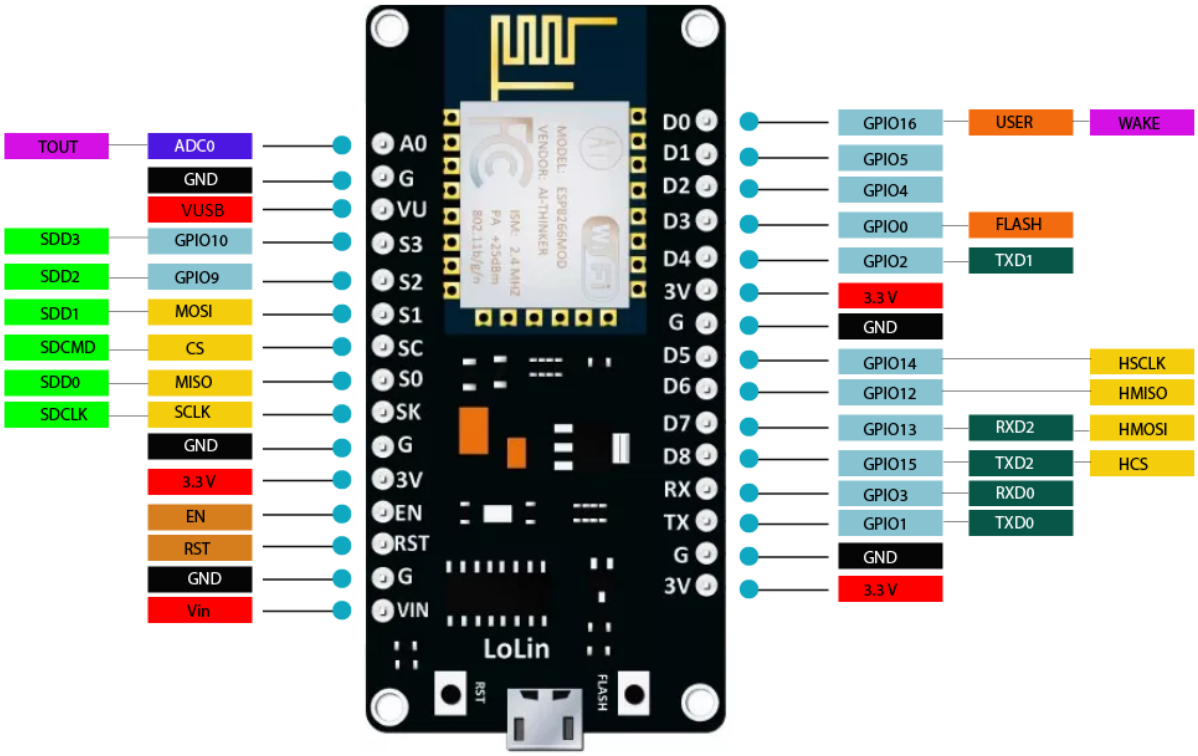

Figura 3.1: Pinout NodeMCU V3

3.1.1 Microcontrolador ESP8266

El **microcontrolador ESP8266** se ha convertido en uno de los *SoC (System on Chip)* más populares en el sector del *Internet de las Cosas*, desde su lanzamiento en el año 2014 por la empresa china Espressif Systems, numerosos productos comerciales lo han incorporado como base para dispositivos inteligentes, y existen en internet gran cantidad de soluciones profesionales y *maker* que utilizan este microcontrolador como base para sus proyectos.

Algunas de las características que han garantizado su éxito son:

- Conexión WiFi con stack TCP/IP completo.
- Programación desde el entorno de Arduino.
- Muy bajo coste, entre 1-3 € dependiendo del modelo.
- Diferentes versiones con mejoras en memoria y número de GPIOs.
- Lo incorporan una gran variedad de kits de desarrollo.

El modelo que incorpora la placa de desarrollo **NodeMCU V3** utilizada para este proyecto es el **ESP-12F** desarrollado por la empresa china *Ai-Thinker Technology* el cual encapsula el chip ESP8266 core.

Los pines que incluye el chip mencionado son los que recoge la tabla 3.1.

El tipo de empaquetado que utiliza es `SMD22` y la memoria flash de la que dispone es de 32 MB (MegaBytes). Para una mayor descripción de las características del chip acuda la ficha técnica del fabricante [5].

Tipo de pin	Numeración	Unidades
GPIO	GPIO0 - GPIO16	16
UART	TX0 - RX0	1 (2 pines)
ADC (10bit)	A0	1
SPI	GPIO12(HMISO), GPIO13(HMOSI), GPIO14(HSCLK), GPIO15(HCS)	1 (4 pines)
RESET	RST	1
ENABLE	EN	1
POWER	GND-VCC	1 (2 pines)

Tabla 3.1: Pinout ESP-12F

3.2 Esquema electrónico de conexionado

La configuración de hardware necesaria para llevar a cabo el proyecto de sensor de CO2 inteligente, requiere la utilización de los siguientes módulos hardware interconectados:

- Placa de desarrollo NodeMCU V3
- Sensor de CO2 modelo Winsen MH-Z19c
- Módulo de alimentación MB-102

La utilización del módulo de alimentación, aunque no es imprescindible para este proyecto al contar con el pin `VU` se ha decidido utilizar para aportar una mayor conectividad en la alimentación (interfaz USB y jack) y la disponibilidad de un botón para encender/apagar la alimentación. Además este módulo permite salida de voltaje a 5V y 3,3V por lo que es compatible con todo tipo de sensores y actuadores que puede manejar el microcontrolador ESP8266.

El esquema de conexionado de los componentes sería el indicado por la figura 3.2

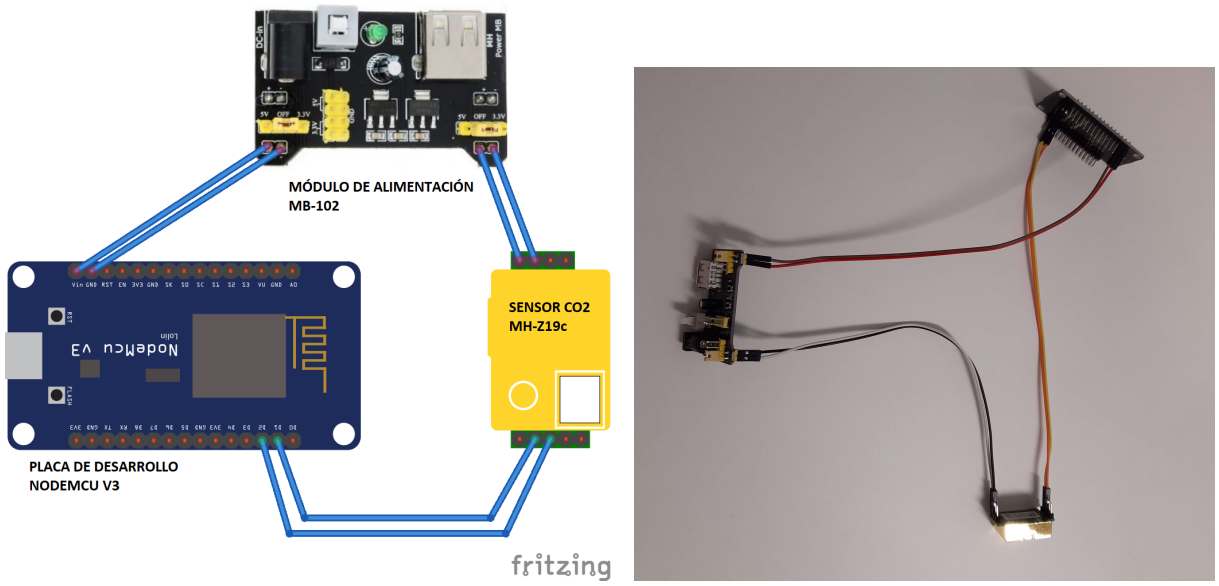

Figura 3.2: Esquema electrónico de conexionado

El módulo de alimentación presenta dos salidas configurables a 5V o 3,3V mediante un conector jumper, en este caso se configurarán las dos a 5V pues el regulador de voltaje de la placa NodeMCU

requiere como mínimo 5V y la alimentación por el pin `Vin` se realiza a través de dicho regulador. Por tanto el conexionado queda como indica la tabla 3.2.

Módulo	Pines	Módulo	Pines
MB-102	+, - (5V)	NodeMCU	Vin, GND
MB-102	+, - (5V)	MH-Z19c	Vin, GND
NodeMCU	D1(TX), D2(RX)	MH-Z19c	RX, TX

Tabla 3.2: Conexionado de pines

3.3 Programación con Arduino

3.3.1 Funcionamiento

La programación del microcontrolador se llevará a cabo mediante el framework de Arduino, pues el core del microcontrolador que se ha utilizado, el Espressif ESP8266, ha sido portado a Arduino con una compatibilidad muy alta en cuanto a librerías y funcionalidades en relación con el Arduino Core. Por ello, las facilidades de programación de este microcontrolador lo han convertido en todo un referente para el desarrollo de dispositivos en el sector del *Internet de las cosas*.

Para la programación que se ha llevado a cabo en este prototipo de sensor de CO2 y Temperatura inteligente se ha tenido en cuenta la inclusión de las siguientes funcionalidades:

- Comunicación serie con el sensor mediante UART.
- Comunicación serie con el PC para depuración de código.
- Configuración dinámica de las credenciales WiFi.
- Función de autocalibración del sensor activada.
- Conexión con broker MQTT y reconexión en caso de pérdida de la comunicación.
- Toma de medidas y envío de datos por MQTT en bucle cada tiempo de muestreo.

La figura 3.3 muestra un diagrama de flujo del programa co2-sensor.ino.

3.3.2 Librería WiFiManager

Uno de los problemas que presentan la programación del microcontrolador ESP8266 es la configuración de las credenciales de acceso a internet, de conexión WiFi con el router, habitualmente esta configuración suele establecerse directamente en el propio fichero del programa con el consecuente inconveniente de que en caso de modificarse el ssid o la contraseña de la red WiFi, o simplemente de mover el microcontrolador a un lugar con otra red, es necesario volver a flashear el programa con las nuevas credenciales de conexión WiFi.

Para solucionar el problema mencionado, y con el objetivo de aproximarnos mejor a un producto comercial, se implementará en el programa la librería *WiFiManager*[10] que permite la configuración dinámica de las credenciales WiFi sin modificar el programa.

Los pasos que sigue la librería para establecer la conexión WiFi son:

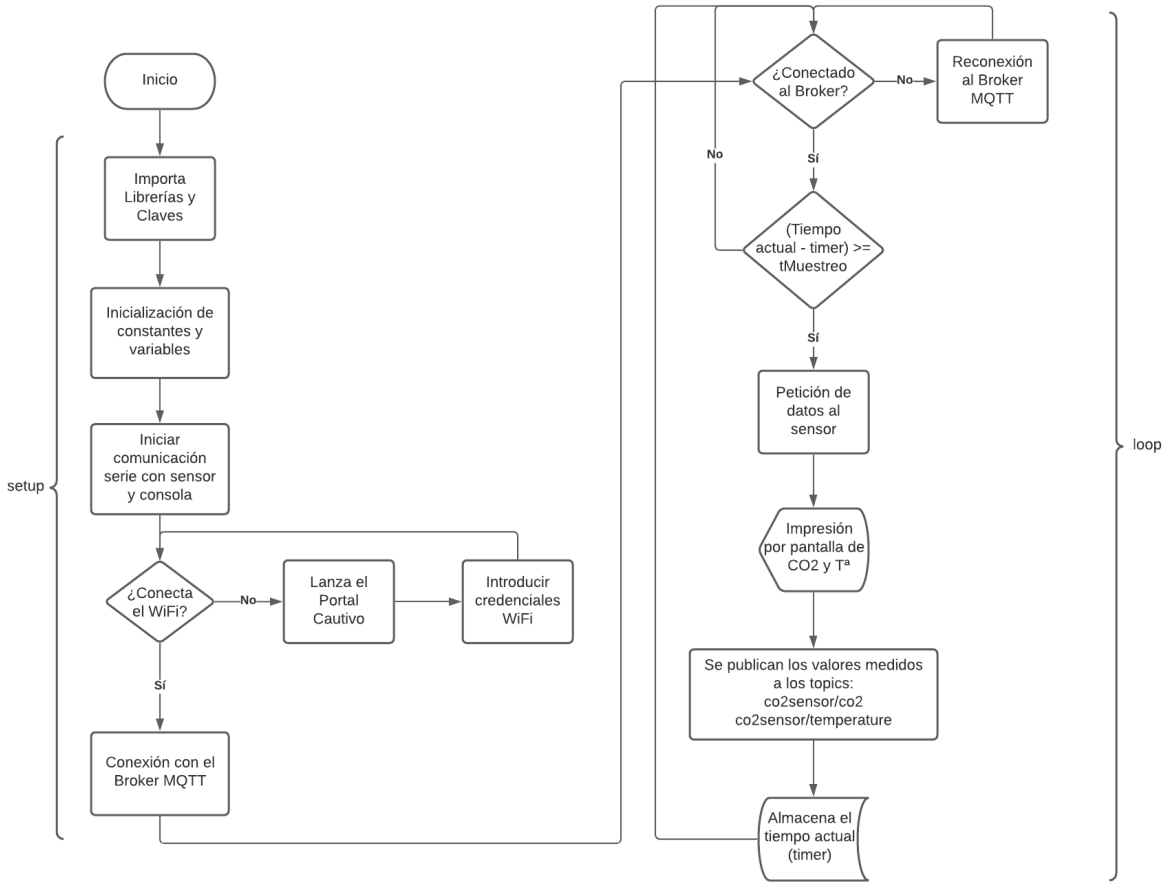

Figura 3.3: Diagrama de Flujo del Programa *Smart CO2 Sensor*

1. Comprueba que se puede realizar la conexión WiFi con las credenciales almacenadas en la memoria EEPROM.

2. Si la conexión no es posible, levanta un portal cautivo en la dirección `192.168.4.1` y con el ssid de la WiFi generada que se haya indicado en el código, en este caso *Smart-CO2-Sensor*.

3. Al acceder a la dirección indicada, el portal cautivo mostrará una página web que permitirá introducir las credenciales WiFi de la nueva red, como muestra la figura 3.4.

4. Tras introducir las credenciales correctamente el microcontrolador las almacenará en la memoria EEPROM y se reiniciará, conectándose en el próximo arranque a la red WiFi que hayamos indicado.

De esta forma puede implementarse una solución sencilla al problema antes comentado, la implementación de un portal cautivo es una forma de comunicación con el usuario que utilizan una gran cantidad de productos comerciales en el ámbito de la **domótica IoT**. En productos comerciales terminados permite además de configurar las credenciales WiFi, configurar otros parámetros como podría ser las configuraciones MQTT, o el tiempo de muestreo del sensor en el caso del producto de este proyecto.

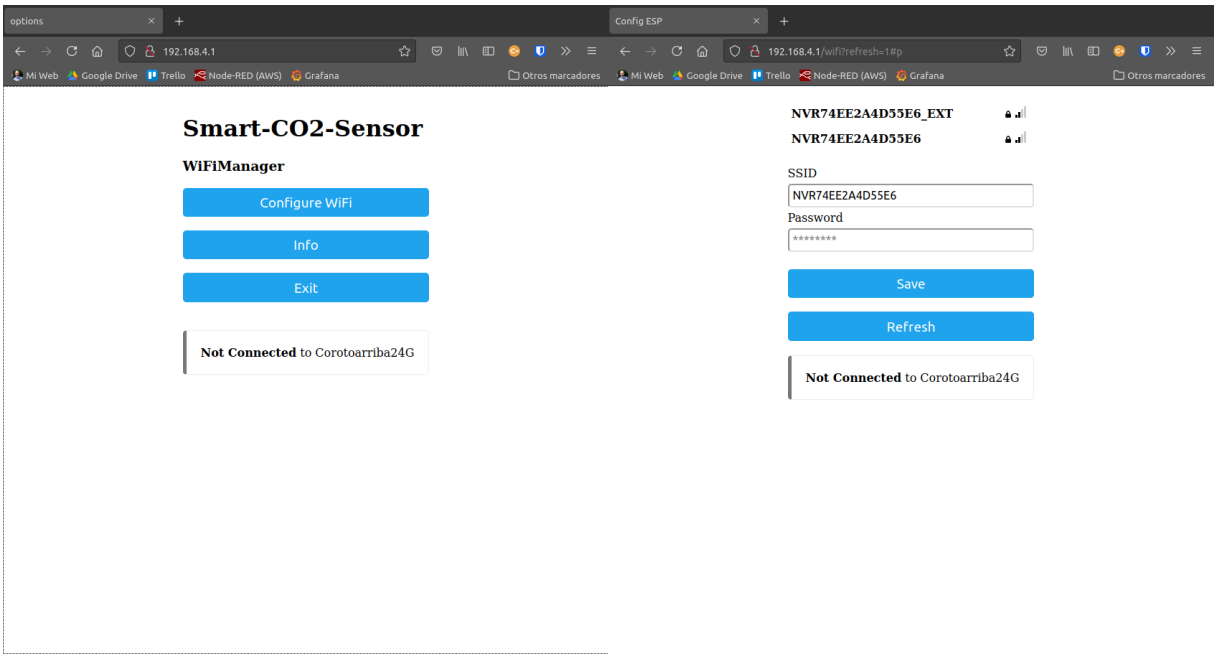

Figura 3.4: Portal Cautivo WiFiManager

3.3.3 Comunicación UART con el sensor

La comunicación entre el microcontrolador y el sensor para obtener los valores registrado por él durante la medida, concentración de CO2 y temperatura, se llevará a cabo mediante el protocolo de comunicación serie UART.

UART son las siglas de *Universal Asynchronous Receiver Transmitter*, se trata de un protocolo de tipo asíncrono, es decir no se comparte la frecuencia de reloj a través del bus de comunicaciones si no que emisor y receptor deben conocer de antemano la velocidad de transmisión de la información en el bus, además se trata de un protocolo serie por lo que los bits de datos se transmiten de forma secuencial a través de un único cable de datos, además el tipo de comunicación es *half-duplex* por lo que la comunicación aunque es bidireccional solo puede darse en un sentido al mismo tiempo, los cables que conformarán la transmisión son llamados RX y TX y deben cruzarse en la conexión entre los diferentes nodos, como muestra la figura 3.5, una descripción más detallada de la arquitectura de funcionamiento de este protocolo puede encontrarse en [4].

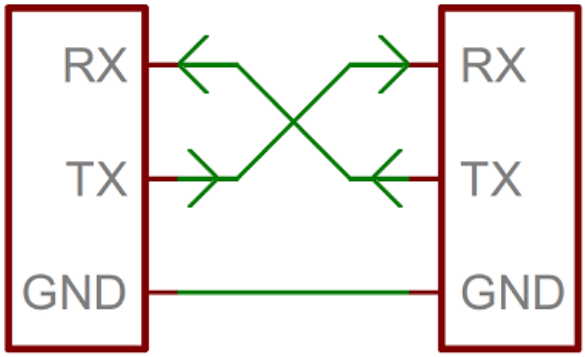

Figura 3.5: Conexionado bus UART

Para una mayor facilidad en la comunicación entre el microcontrolador y el sensor se utilizará la

librería MH-Z19 [3], cuya *API* implementa una interfaz de comunicación más sencilla, en lugar de establecer la comunicación de forma directa con el sensor mediante UART.

De esta forma, como se puede leer en el código de Arduino mencionado en 3.3.1 se utilizarán los pines 4 y 5 de la placa NodeMCU como pines RX y TX respectivamente, para ello se hará uso de la librería *SoftwareSerial* incluida en el core de Arduino. Esta librería permite establecer una comunicación serie mediante software, utilizando otros pines que no hayan sido configurados mediante hardware para utilizar la comunicación UART, la razón de realizar la comunicación de esta forma es la de poder disponer de la comunicación hardware UART de la placa para comunicar con la salida por consola al ordenador mediante el USB y así realizar la depuración del código programado.

La velocidad de comunicación a la que el microcontrolador se comunica con el sensor se ha establecido en 9600 bps y la frecuencia de muestreo se ve limitada por la capacidad de respuesta del sensor, de la ficha técnica del fabricante se obtiene que el tiempo de respuesta del sensor ante cambios en el valor medido es de 120 segundos por lo que la frecuencia de muestreo que seleccionemos debe ser mayor o igual a este valor, por lo que se selecciona un **tiempo de muestreo de 120 segundos** para la lectura de los valores de concentración de CO2 y Temperatura.

3.3.4 Envío de datos por MQTT

En el campo del *Internet de las cosas* uno de los protocolos más utilizados para transmitir información entre dispositivos es el protocolo **MQTT**, se trata de un protocolo de tipo publicación/suscripción, los diferentes dispositivos se suscriben y publican mensajes a un broker central que es el que gestiona los diferentes clientes y los paquetes que se envían. Se trata de un protocolo de envío de mensajes muy ligero por lo que es ideal para dispositivos *IoT* de bajos recursos y la comunicación se establece sobre la pila TCP/IP. Una información más extensa sobre el funcionamiento de este protocolo se puede encontrar en la página web oficial de la organización[7].

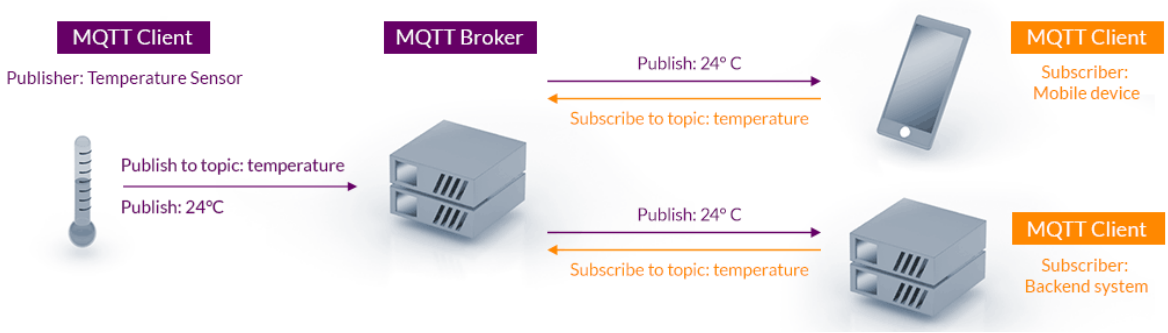

Figura 3.6: Comunicación MQTT

Tomando la figura 3.6 como ejemplo para la transmisión de información de un sensor, el caso del sensor de CO2 de este proyecto sigue el mismo patrón, el sensor registrará los valores de concentración de CO2 y de Temperatura cuando se lo pida el microcontrolador y éste los publicará al **broker MQTT** instalado en el servidor, los topics (canales utilizados para identificar la información publicada) empleados para el sensor son `co2sensor/co2` y `co2sensor/temperature` respectivamente, como se explicará en el capítulo 4.2.1, el **Backend** del servidor se suscribirá también al broker MQTT para recibir el valor de la concentración de CO2 y la Temperatura y los almacenará en la base de datos.

La programación de la comunicación MQTT mediante el entorno Arduino se puede realizar con diferentes librerías, pero la más utilizada es la librería PubSubClient[2] que es la que se ha implementado

para el sensor de CO2 de este proyecto. Para establecer la comunicación con el broker es necesario conocer la dirección IP, el puerto y el usuario y contraseña configurados para el broker, ésta configuración se ha de incluir en el script que ejecutará el microcontrolador, para evitar introducir las credenciales en plano en el fichero principal, se crea un fichero `keys.h` con las mismas, el cual no se publicará en el repositorio, dicho fichero será importado en el script principal, co2-sensor.ino, utilizando la siguiente instrucción.

```
#include "keys.h"
```

La función `callback`, encargada de ejecutar acciones al recibir un mensaje en los topics suscritos, no se ha implementado pues en este prototipo sólo se busca que el sensor mida los valores de concentración de CO2 y de Temperatura buscados. En un producto final podría implementarse dicha funcionalidad y que el sensor se suscriba, por ejemplo, a un topic de configuración donde se le puedan cambiar ciertos parámetros de la configuración a través de MQTT, como podría ser el tiempo de muestreo.

3.4 Carcasa impresa en 3D

Con el objetivo de albergar todos los componentes electrónicos y dar una primera aproximación al aspecto de un producto final, se ha modelado una carcasa en tres dimensiones adaptada a las características de los 3 módulos que componen el sistema, el módulo de alimentación, el microcontrolador NodeMCU V3, el sensor de CO2 MH-Z19C y los cables dupont necesarios para las conexiones entre ellos.

Entre las características a tener en cuenta para el diseño de la carcasa están:

- Las dimensiones de cada módulo.
- La posible ubicación de los diferentes módulos en la carcasa
- Sistemas de sujeción de los módulos, como pestañas o pivotes a medida para el asiento de las placas o el ajuste a presión del sensor.
- Ranuras de ventilación próximas al sensor con el objetivo de que el aire fluya con el menor impedimento posible al interior del sensor y no interfiera en los valores medidos.
- Agujeros para la salida de los conectores del módulo de alimentación y su conexión al cable del transformador sin abrir la carcasa, además de para el accionamiento del botón ON/OFF del módulo de alimentación.

Para el diseño de la carcasa se ha utilizado el software Rhinoceros 3D y se ha generado el archivo de modelado con la extensión `.3dm` además de los ficheros `.stl` para su impresión, dichos ficheros se encuentran adjuntos en el repositorio del proyecto [8].

La impresora 3d utilizada para su impresión ha sido la **Creality Ender 3 Pro**, a continuación la figura 3.7 muestra unas imágenes con la carcasa modelada y el resultado obtenido.

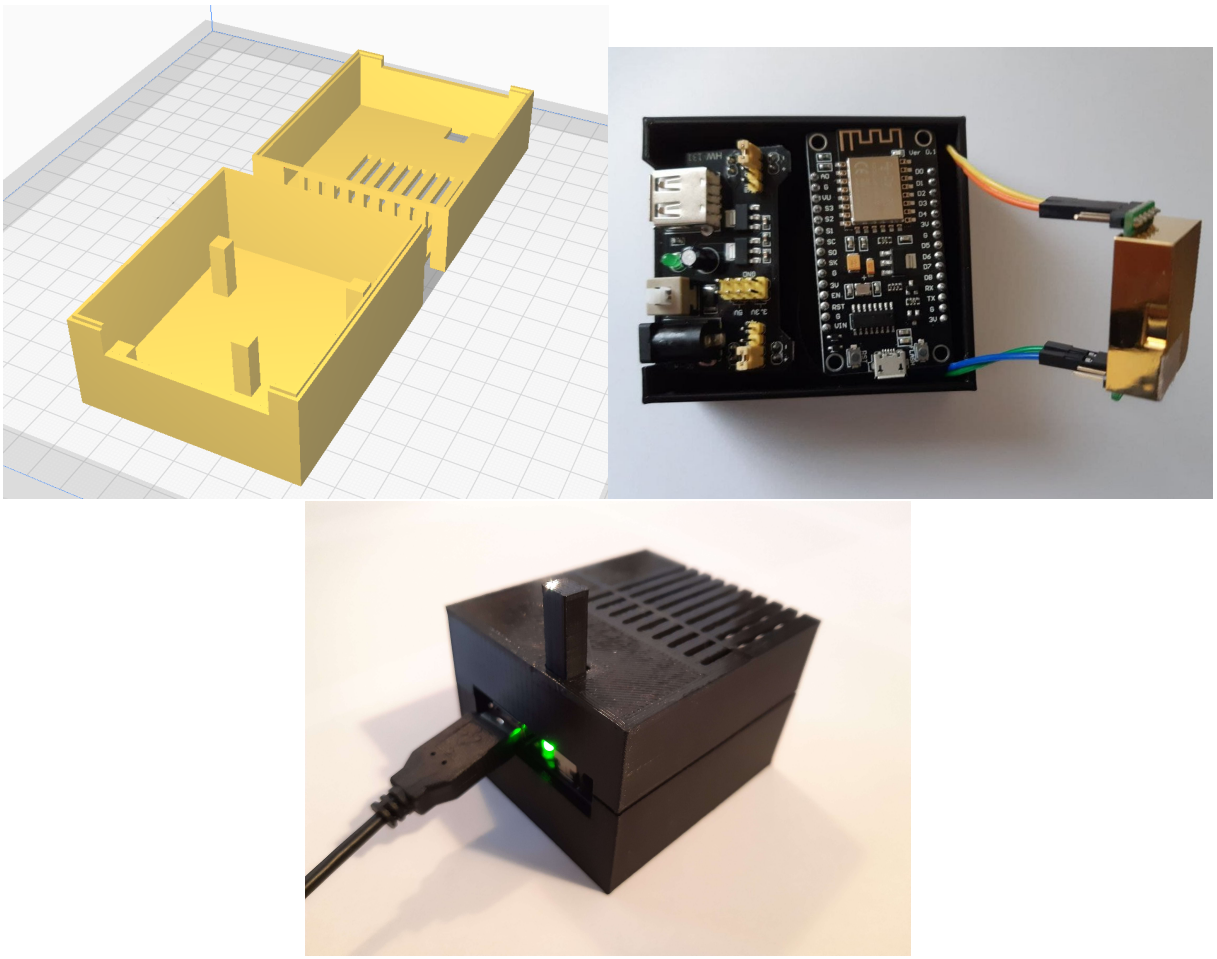

Figura 3.7: Carcasa impresa en 3D

4 Arquitectura Cloud

4.1 Instancia en servidor Amazon AWS

La recepción, almacenamiento y tratamiento de los datos enviados por el sensor de CO2 se llevará a cabo en una instancia de máquina virtual haciendo uso de los servicios en la nube que la empresa Amazon con su plataforma AWS ofrece para particulares y empresas.

En concreto para este proyecto de TFM se utilizará una instancia **Amazon EC2** utilizando la capa gratuita que proporciona durante el primer año de uso de sus servicios, esto no supone un problema pues el servicio Amazon EC2 *(Elastic Compute Cloud)* permite el redimensionado de las características técnicas de la máquina virtual de forma elástica a medida que aumente la demanda y el tráfico al servidor.

La capa gratuita permitirá ejecutar una instancia de tipo `t2.micro` con 1 CPU virtual, 1 GB de memoria RAM y almacenamiento elástico EBS.

La distribución a instalar para esta instancia será **Ubuntu Server 20.04** y se accederá a ella para el control remoto e instalación de los servicios mediante una conexión remota `ssh` utilizando el par de claves pública/privada obtenido durante la creación de la instancia.

Por último, será necesario establecer la apertura de puertos a internet de los servicios que se necesiten. Como se verá en la sección 4.2 los puertos a abrir serán el 9000 para la escucha del broker MQTT Mosquitto y los puertos 80 y 443 para las peticiones http y https al proxy inverso.

4.2 Despliegue de servicios

4.2.1 Arquitectura

La arquitectura a desplegar para la recogida, tratamiento y visualización de los datos enviados por el sensor se llevará a cabo bajo el entorno de virtualización en contenedores de Docker, lo que permite realizar un despliegue automático de todos los servicios con las configuraciones requeridas, como se explicará en el apartado 4.2.2.

Los servicios a desplegar en el lado del servidor, todos ellos de software libre, son:

- **Backend**, comprende la parte de conexión entre la recepción de la información del broker MQTT y el almacenamiento de los datos en la BBDD, ver sección 4.3, se utilizará el servicio de programación de flujos gráficos de información *Node-Red*.
- **Base de datos**, almacena la información de la concentración de CO2 y de la temperatura como series temporales de tiempo, `timestamp - valor`, se utilizará la base de datos de series temporales *InfluxDB*, ver sección 4.4.
- **Frontend**, comprende la parte de visualización gráfica de los valores registrados en la base de datos, se utilizará el servicio *Grafana* que permite mediante consultas a las BBDD obtener los valores del sensor en una determinada ventana de tiempo y mostrar los valores en diferentes tipos de dashboards, ver sección 4.5.
- **Broker MQTT**, se desplegará el broker *Eclipse Mosquitto* para que el sensor pueda publicar los datos a él de forma securizada con usuario y contraseña de conexión al broker.
- **Proxy inverso**, será el encargado de redirigir el tráfico entrante ante peticiones http y https al servicio escogido en función del subdominio indicado en la url, esto permite además de balancear el tráfico, abrir varios servicios web a internet sin necesidad de la apertura de puertos adicionales con el consecuente riesgo que supondría para el servidor. El proxy inverso a utilizar será *Caddy* que además permite de forma automatizada la generación de los certificados https mediante la autoridad de certificación gratuita *Let's Encrypt*.

La siguiente figura 4.1 muestra un esquema de la arquitectura de servicios que conformará el servidor.

4.2.2 Docker

Docker es un proyecto de código abierto que automatiza el despliegue de aplicaciones dentro de contenedores de software, proporcionando una capa adicional de abstracción y automatización de virtualización de aplicaciones en múltiples sistemas operativos.

Un contenedor Docker, a diferencia de una máquina virtual, no requiere incluir un sistema operativo independiente. En su lugar, se basa en las funcionalidades del kernel y utiliza el aislamiento de recursos (CPU, la memoria, el bloque E/S, red, etc.) y namespaces separados para aislar la vista de una

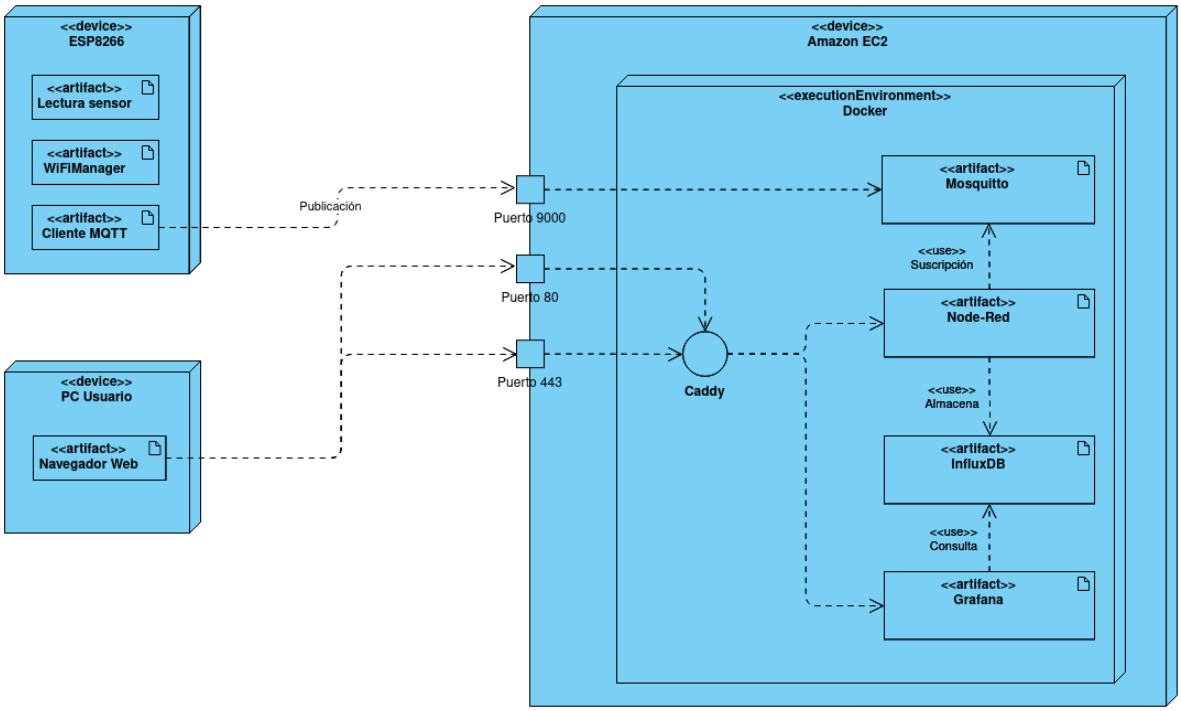

Figura 4.1: Arquitectura Cloud - Diagrama de despliegue

aplicación del sistema operativo. Docker accede a la virtualización del kernel Linux ya sea directamente a través de la biblioteca libcontainer, o indirectamente a través de libvirt, LXC o systemd-nspawn.

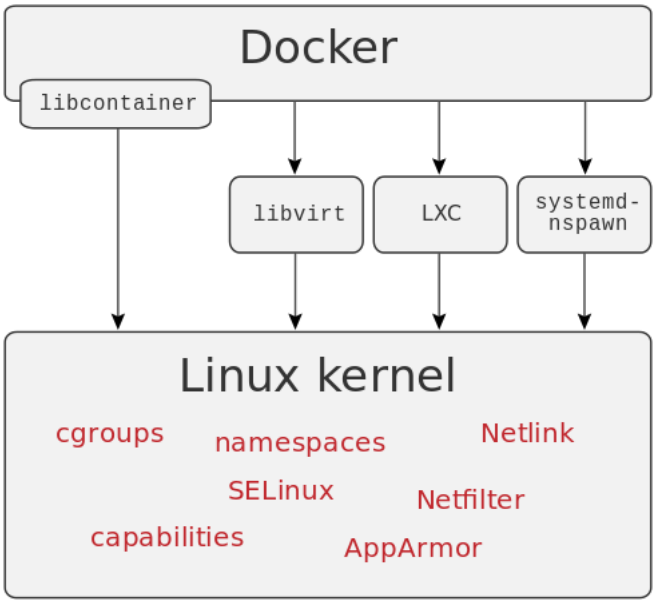

Figura 4.2: Interfaces Docker

Existen diferentes tecnologías de contenerización en el mercado como **Docker** o **Podman**, aunque por su popularidad y mayor cantidad de documentación y de imágenes en repositorios públicos de la comunidad se decidirá utilizar Docker.

Los beneficios que se obtienen de implementar esta tecnología en un servidor de producción son:

- **Automatización de los despliegues de las aplicaciones**, haciendo uso de ficheros *Dockerfile* para crear la imagen y de ficheros *docker-compose.yaml* para el despliegue de todos los contenedores necesarios y variables de entorno que requiera la aplicación.
- **Encapsulación de versiones y dependencias**, permite tener imágenes de diferentes versiones de la aplicación, por lo que si ocurre algún fallo en producción con la última versión desplegada, simplemente es necesario levantar el contenedor de la versión anterior para tenerla operativa en segundos. Además al estar encapsulados las diferentes dependencias de cada contenedor permite utilizar diferentes versiones de python, apache, php o demás servicios en la misma máquina sin riesgo de conflictos entre versiones.
- **Mayor seguridad**, el código y los ficheros de configuración son encapsulados en la imagen de docker por lo que supone una barrera de entrada adicional ante cualquier posible manipulación del mismo durante un ataque informático.
- **Mejor escalabilidad**, el uso de contenedores permite una mejor segmentación del código de la aplicación y su orientación hacia microservicios, un tipo de arquitectura que permite una mejor escalabilidad para aplicaciones grandes.
- **Mejor portabilidad**, pudiendo fácilmente trasladar las aplicaciones de un servidor a otro sin volver a instalar todo, simplemente creando un nuevo contenedor a partir de la imagen.
- **Permite la implementación con Kubernetes**, una de las ventajas de usar kubernetes junto con contenedores es la posibilidad de alojarlos en diferentes máquinas, lo que brinda la posibilidad de utilizar una arquitectura de computación distribuida, ideal para aplicaciones con mucho tráfico y gran cantidad de datos a procesar *(Big Data)*.

En general, las aplicaciones que utilizan la tecnología de contenerización con Docker emplean una arquitectura como la que describe la figura 4.3.

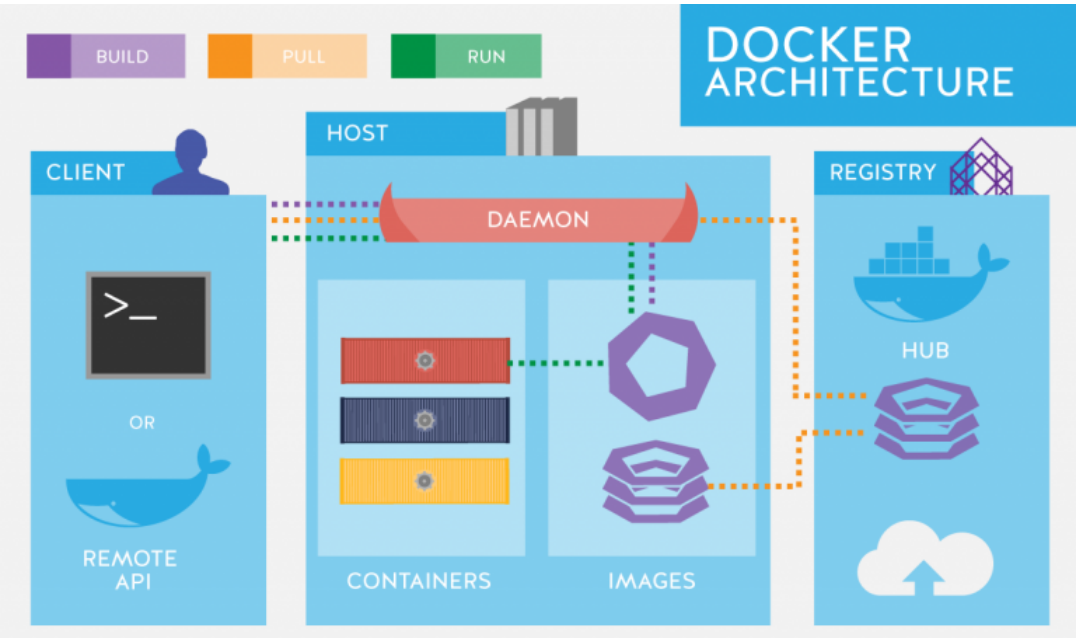

Figura 4.3: Arquitectura Docker

Dadas las numerosas ventajas que ofrece el uso de aplicaciones contenerizadas utilizando Docker, y dado que es hacia donde se está moviendo, con mucha aceptación actualmente, el mercado del

desarrollo de software, lo más conveniente es utilizar este tipo de tecnología para el despliegue de los servicios de los que hará uso el sensor de CO2 inteligente, mejorando la escalabilidad y la portabilidad del proyecto para futuras versiones y mejoras del producto.

Para la aplicación actual se hará uso de servicios de software libre, por lo que no se contempla la programación a medida de una plataforma cloud para la gestión de la información enviada por el sensor.

Se desplegarán los servicios mediante la descripción de los contenedores a través del fichero docker-compose.yaml.

En este fichero se describen las imágenes a descargar del repositorio de acceso público *Docker HUB*. Será necesario indicar los puertos que queremos que el contenedor exponga a la máquina, como por ejemplo el puerto 9000 para el broker MQTT. Se establecerán las variables de entorno que utilizaran los contenedores y que permiten realizar determinadas configuraciones como establecer usuarios y contraseñas a los diferentes servicios, las variables de entorno se escribirán en un fichero oculto asociado `.env` para mayor seguridad. Por último también se establecen los volúmenes a compartir entre los contenedores y la máquina host, para asegurar la persistencia de los datos y un fácil acceso a los mismos.

Una vez finalizada la configuración del despliegue será necesario ejecutar el siguiente comando en una terminal para levantar todos los contenedores con funcionamiento en segundo plano *(detached)*.

```
sudo docker-compose up -d
```

En este momento Docker descargará las imágenes de *Docker HUB* y levantará los contenedores con las configuraciones indicadas, se puede comprobar el estado de los mismos mediante el siguiente comando.

```
sudo docker ps
```

4.3 Backend con Node-Red

Se entiende por **Backend** la parte de un servicio, como una página web, encargada de la lógica y que suele llevarse a cabo en un servidor.

En el caso del sistema *IoT* de monitorización de la calidad del aire que es objeto de este TFM, el backend será el encargado de recoger todos los datos del sensor (CO2 y Temperatura) que llegan a través del broker MQTT, hacer las transformaciones necesarias según el modelo de datos de la base de datos y almacenarlos en ésta.

Como **Backend** para este sistema se ha decidido utilizar el servicio de código libre *Node-Red* que es una herramienta de programación visual basada en diagramas de flujo y desarrollada por la empresa **IBM**.

Con este tipo de programación pueden conectarse diferentes dispositivos hardware IoT de forma sencilla, con otros servicios web que expongan su API de forma pública. Además, *Node-Red* cuenta con una gran cantidad de nodos creados por numerosos desarrolladores que permiten su integración directa con otros servicios conocidos, como los broker MQTT o conectores para bases de datos.

La siguiente figura 4.4 muestra el diagrama de flujo realizado para este sistema de monitorización de la calidad del aire.

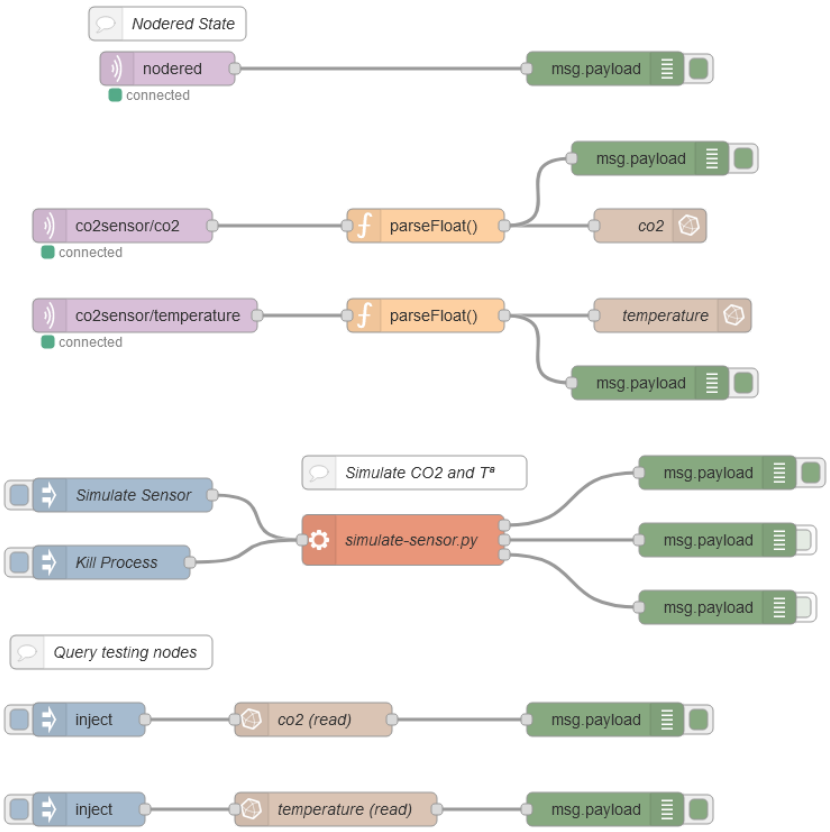

Figura 4.4: Diagrama de flujo en Node-Red

A continuación se describirá cada flujo de forma detallada comenzando de arriba a abajo.

- **Flujo de estado**, se suscribe al broker MQTT y lee el topic `nodered` que se ha configurado para publicar el estado de conexión del cliente Node-Red al broker MQTT, el mensaje del topic lo envía a un nodo debug para mostrarlo por consola.
- **Flujo de lectura/escritura de CO2**, se suscribe al topic `co2sensor/co2` en el cual se publican los valores de concentración de CO2 en ppm, el payload recibido es *parseado* de tipo String a tipo Float y posteriormente es enviado al nodo de la base de datos *InfluxDB* que guardará ese valor en el *bucket (equivalente a base de datos)* llamado `smart-co2-sensor` y como medición `co2`, además el payload convertido a float es enviado a un nodo debug para su visionado por consola.
- **Flujo de lectura/escritura de Temperatura**, se trata de un flujo similar al anterior en funcionamiento, pero en este caso el topic a suscribirse es `co2sensor/temperature` y se escribirá en la BBDD en el mismo bucket pero como medición `temperature`.
- **Flujo de simulación**, se trata de un flujo que activa de forma manual mediante los nodos input un script desarrollado en python, simulate-sensor.py, que se encarga de generar de forma aleatoria mediante una distribución normal valores de temperatura y concentración de CO2 y publicarlos al broker MQTT, de esta forma puede comprobarse el funcionamiento de los servicios del servidor de forma aislada al funcionamiento y datos que envía el sensor real.
- **Flujo de consulta de CO2**, se trata de un flujo que se activa de forma manual mediante un nodo input y activará un nodo de consulta de la base de datos que devolverá los valores de la

última hora de concentración de CO2 y los mostrará por consola, útil para comprobar el correcto almacenamiento de los valores en la base de datos.

- **Flujo de consulta de Temperatura**, se trata de un flujo similar al anterior que devuelve los valores de temperatura de la última hora y los muestra por consola.

Los flujos antes mencionados constituyen la lógica del servidor y permiten hacer las operaciones de mantenimiento y búsqueda de errores mediante los flujos de simulación y consulta.

4.4 BBDD de series temporales

Dadas las características de los datos a almacenar, valores fluctuantes con el tiempo, se ha decidido utilizar un tipo de base de datos diseñada específicamente para tal fin en lugar de usar una base de datos de tipo relacional SQL.

Entre las ventajas que supone utilizar este tipo de bases de datos frente a las bases de datos relacionales están las siguientes:

- **Mejor escalabilidad**, las bases de datos NoSQL de series temporales tienen una mejor escalabilidad al utilizar almacenamiento de datos de tipo clave-valor en lugar de tablas estructuradas.
- **Mejor velocidad**, la inserción de datos es más eficiente y rápida en este tipo de bases de datos lo que permitiría almacenar más datos por segundo, aumentando la frecuencia de muestreo.
- **Esquemas dinámicos**, el esquema no tiene que ser predefinido previamente durante el diseño de la base de datos, es posible por ejemplo añadir un nuevo sensor y que utilice un esquema de valores diferente más adaptado al mismo.

La base de datos escogida para este proyecto ha sido **InfluxDB** una base de datos de código abierto de tipo series temporales diseñada especificamente para aplicaciones del *Internet of Things* y que presenta un gran número de conectores para diferentes lenguajes de programación como Javascript o Python además de otros servicios de visualización de datos como **AWS Cloudwatch** o **Grafana**.

Utiliza un lenguaje de consulta de tipo scripting denominado *Flux* inspirado en Javascript. En este tipo de base de datos, los datos se almacenan en unas estructuras llamadas *buckets* con estructura del tipo clave-valor.

Las consultas utilizadas para la visualización de los datos, mediante el lenguaje *Flux* siguen una estructura similar a la siguiente:

```
  from(bucket: "smart-co2-sensor")
|> range(start: -1h)
|> filter(fn: (r) =>
  r._measurement == "co2"
)
```

Esta consulta devolverá los valores de ppm de CO2 junto con su índice de tiempo *(timestamp)* de la última hora desde el momento en que se ejecuta la consulta.

4.5 Frontend

Se entiende por **Frontend** la parte de un servicio, como una página web, encargada de la interfaz gráfica de usuario, es la parte visual de todo el servicio y debe garantizar la simplicidad de uso suficiente para una correcta interacción con el usuario.

El sistema planteado en este trabajo consta de dos interfaces de interacción con el usuario, una interfaz de visualización de dashboards con diferentes gráficas para los valores de concetración de CO2 y de Temperatura, y la recepción de notificaciones a través del chat de mensajería Telegram.

4.5.1 Dashboards de visualización con Grafana

El servicio de visualización de dashboards de código libre **Grafana** permite la visualización de datos y métricas a través de cuadros de mando mediante consultas a múltiples fuentes (bases de datos).

Para el presente proyecto se creará un dashboard denominado `Smart CO2 Sensor Dashboard` que se puede observar en la figura 4.5.

Figura 4.5: Dashboard en Grafana

Se pueden visualizar los datos de la última hora del dashboard en tiempo real accediendo al citado enlace [1].

El dashboard está formado por cuatro paneles de mando, a la izquierda se muestra la visualización de las gráficas temporales de la concentración de CO2 en partes por millón *(ppm)* y a la gráfica de variación de la temperatura en grados *Celsius(ºC)*. A la derecha se muestra un medidor en tiempo real de la concentración de CO2 y un panel de alertas sobre dicha concentración.

La alerta está configurada para disparar cuando se sobrepasen los 1000 ppm de concentración de CO2 medidos por el sensor como valor medio en los últimos 5 minutos. Esta alerta cambiará el color de la notificación en el panel de alertas y el gráfico pasará a tener fondo rojizo hasta volver a la normalidad (por debajo de 1000 ppm). Además, se enviará la notificación a través de un bot de notificaciones en

Telegram, como se explica en el apartado 4.5.2.

4.5.2 Bot de notificaciones en Telegram

Una de las funcionalidades que incorpora de forma integrada **Grafana** es la de enviar las alertas mediante un bot de mensajería de la aplicación **Telegram**, algo muy útil pues notificará en todo momento en un chat personal o un chat grupal a las personas de que el nivel de CO2 ha excedido el máximo recomendado por lo que no será necesario que alguien se encuentre observando el dashboard de **Grafana** para percatarse de dicha alerta.

De esta forma se puede asegurar una rápida intervención, como la apertura de ventanas en la sala, para evitar que los niveles de concentración de CO2 continúen en aumento.

Para configurar el bot de notificaciones será necesario crear un nuevo bot mediante la utilidad *Bot Father* implementada por defecto en **Telegram**, solicitaremos la creación de un nuevo bot y obtendremos un **API TOKEN** que deberemos introducir en la configuración de **Grafana** de dicho bot, además deberemos introducir el id de usuario de **Telegram** o del chat grupal donde queramos que se envíen las alertas, los bots son públicos en este chat de mensajería pero limitando su uso mediante el id mencionado, sólo podrán interactuar con ellos el usuario con esa id o el chat grupal en caso de ser un id de chat grupal.

La figura 4.6 muestra una imagen de la recepción de las alertas en este chat de mensajería.

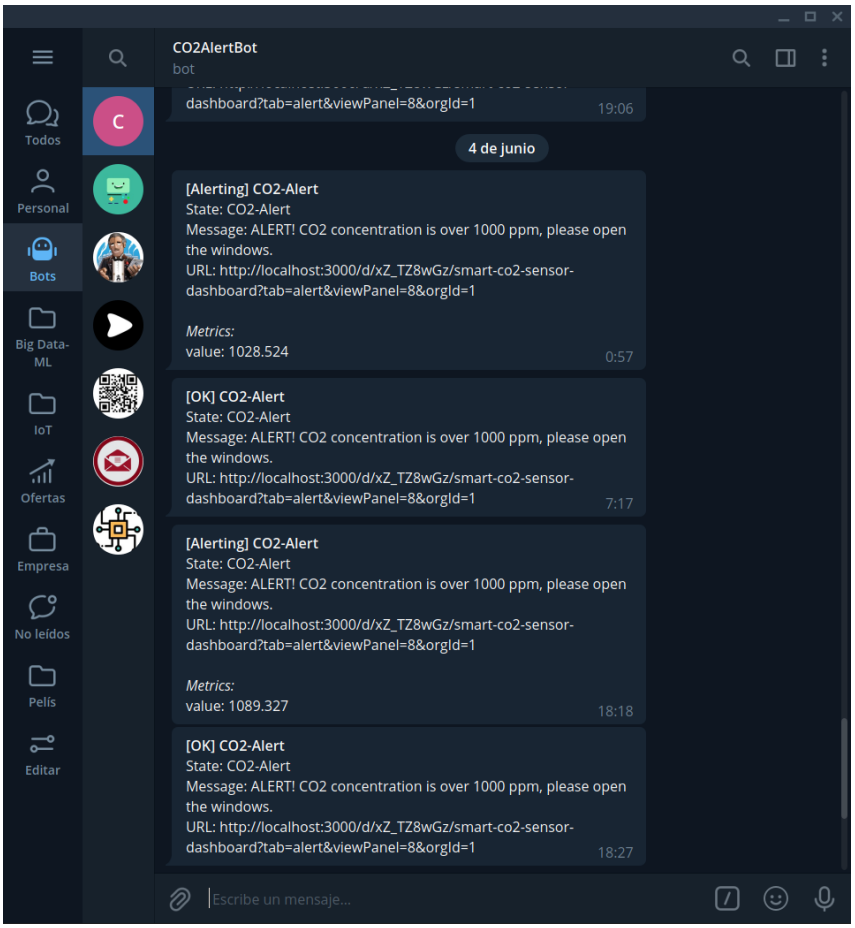

Figura 4.6: Bot de alertas en Telegram

5 Conclusiones y propuestas de mejora

5.1 Conclusiones y resultados obtenidos

Los resultados obtenidos de este proyecto de sensor de CO2 inteligente permiten demostrar la posibilidad de desarrollar un sistema de monitorización de la calidad del aire de bajo coste utilizando tecnologías de software y hardware libre.

Los principales problemas encontrados durante el desarrollo del proyecto han sido:

- Los plazos de entrega del sensor de CO2, al ser un producto de fabricante chino, el plazo de entrega puede ser de hasta 1 semana superior al de sus competidores.
- La medición de temperatura del sensor no es demasiado precisa al inferirla éste mediante la medición directa de la concentración de CO2.
- Debido a que el sensor utiliza tecnología NDIR, la ventilación de la carcasa debe asegurar un correcto flujo de aire en él, limitando el diseño de la misma.
- El elevado consumo tanto del microcontrolador como del sensor hacen inviable una solución alimentada por baterías.

En conclusión, el producto obtenido de este trabajo es un prototipo funcional que muestra como juntando diferentes tecnologías puede hacerse un producto accesible, y que permite mejorar la calidad del aire de locales cerrados mediante la notificación y previsión de la concentración de CO2, evitando con ello la transmisión de enfermedades respiratorias y el cansancio que altos niveles de concentración provocan en las personas.

5.2 Propuestas de mejora

El objetivo de este apartado es proponer una serie de mejoras que permitan en el futuro desarrollar una versión comercial del prototipo diseñado en este Trabajo Fin de Máster y solucionar los problemas detectados en el apartado 5.1.

Entre las mejoras que se recomienda implementar de cara a obtener un producto final están:

- Diseño de una placa PCB a medida.
- Añadir un sensor de temperatura y humedad integrado.
- Rediseñar la carcasa según los estándares estéticos del mercado y teniendo en cuenta las características de ventilación requeridas.
- Asegurar un suministro continuo, fiable y un cierto stock de sensores de CO2 para evitar problemas con los plazos de entrega.
- Mejorar el portal cautivo añadiendo más posibilidades de configuración y una estética más comercial.
- Diseño de una interfaz frontend a medida tanto para plataformas móviles como para el navegador web.

- En caso de optar por integrar alimentación con baterías, utilizar un microcontrolador con protocolo **Bluetooth Low Energy (BLE)**, el volcado de datos a la nube se hará a través de la comunicación con el smartphone que actuará de pasarela a internet.

Estas son algunas de las mejoras que se podrían implementar en el desarrollo de un producto final, sin descuidar otros aspectos menos técnicos como una correcta campaña de marketing y publicidad, un buen servicio de soporte al cliente y un sistema logístico de distribución adecuado.

6 Referencias

[1] Grafana Labs Company. Dashboard smart co2 sensor. https://grafana.smartco2.duckdns.org/d/xZ_TZ8wGz/smart-co2-sensor-dashboard?orgId=1&from=1622722761841&to=1622726361842, 2021. Configurado por David Adrián Rodríguez García (Usuario: demo, Contraseña: demo).

[2] Nick O'Leary et al. Librería pubsubclient para arduino. https://pubsubclient.knolleary.net/, 2009. Librería para la comunicación MQTT utilizando el entorno Arduino.

[3] WifWaf et al. Librería mh-z19 para arduino. https://github.com/WifWaf/MH-Z19, 2019. Librería para microcontroladores Arduino y Espressif para la comunicación con la familia de sensores MH-Z19 de Winsen.

[4] Texas Instruments Inc. *KeyStone Architecture Universal Asynchronous Receiver/Transmitter (UART) User Guide*. Texas Instruments Inc., Dallas, TX, USA, 2010.

[5] Shenzhen Ai-Thinker Technology Co. Ltd. Ficha técnica soc esp-12f. https://docs.ai-thinker.com/_media/esp8266/docs/esp-12f_product_specification_en.pdf, 2018.

[6] Zhengzhou Winsen Electronics Technology Co. Ltd. Ficha técnica sensor de co2 mh-z19c. https://www.winsen-sensor.com/d/files/infrared-gas-sensor/mh-z19c-pins-type-co2-manual-ver1_0.pdf, 2020.

[7] MQTT.org. Web oficial organización mqtt. https://mqtt.org/, 2020.

[8] David Adrián Rodríguez García. Open co2 monitoring system. https://github.com/davidadrianrg/open-co2-monitoring-system, 2021. Proyecto de sistema para la monitorización de CO2 mediante hardware IoT de bajo coste y servicios cloud de código abierto para el tratamiento, almacenamiento y visualización de los datos.

[9] PCE Ibérica S.L. Ficha técnica medidor de co2 pce-cmm5. https://www.pce-instruments.com/espanol/api/getartfile?_fnr=1504204&_dsp=inline.

[10] tzapu et al. Librería wifimanager. https://github.com/tzapu/WiFiManager, 2018. Librería para la gestión de la conexión WiFi en el microcontrolador ESP8266 mediante portal cautivo.

[11] María Cruz Minguillón Xavier Querol José Manuel Felisi y Tomás Garrido. *Guía para ventilación en aulas*. Versión 3. CSIC-IDAEA, Ministerio de Ciencia e Innovación y Mesura, España, 2020.

[12] Miguel Ángel Casanova. Proyecto codos. https://github.com/miguelangelcasanova/codos, 2020. Proyecto CODOS para la monitorización de la calidad del aire con hardware de bajo coste.

Anexos

Listado de materiales

A continuación se adjuntan en la tabla 6.1 el listado de materiales electrónicos, informáticos y software utilizados.

Referencia	**Descripción**	**Cantidad**	**Proveedor**
Sensor MH-Z19c	Sensor de concentración de CO2 con tecnología NDIR	1	Aliexpress - Winsen
NodeMCU V3	Placa de desarrollo IoT con SoC ESP8266	1	Amazon - AZDelivery
MB-102	Módulo de alimentación 3.3V/5V	1	Amazon - AZDelivery
Cable Dupont	Cable para la interconexión entre módulos	6	Amazon - SODIAL
Creality Ender 3 Pro	Impresora 3D	1	Aliexpress - Creality
Plástico PLA Negro	Filamento PLA para la impresión de la carcasa	0,05	Amazon - Geeetech
Amazon EC2	Instancia de servidor t2.micro en Amazon AWS	1	Amazon
Docker	Software de virtualización en contenedores	1	Docker
Node-Red	Software de programación gráfica basada en flujos	1	JSFoundation - IBM
Eclipse Mosquitto	Broker MQTT	1	Eclipse Foundation
InfluxDB	Base de datos de series temporales	1	InfluxData
Grafana	Software para la visualización de paneles de control	1	Grafana Labs
Bot de Telegram	Bot para la notificación de alertas	1	Telegram
Caddy	Proxy inverso con encriptación https automática	1	Stack Holdings

Tabla 6.1: Listado de materiales empleados.

Presupuesto

A continuación se muestra en la tabla 6.2 una estimación del presupuesto del proyecto.

Referencia	**Cantidad**	**Unidad**	**Precio unitario**	**Precio Total**
Sensor MH-Z19c	1	ud	17,09 €	17,09 €
NodeMCU V3	1	ud	7,99 €	7,99 €
MB-102	1	ud	3,99 €	3,99 €
Cable Dupont	6	ud	0,09 €	0,53 €
Creality Ender 3 Pro	1	ud	175,00 €	175,00 €
Plástico PLA Negro	0,05	kg	20,00 €	1,00 €
Amazon EC2 (capa gratuita)	1	ud	- €	- €
Ingeniero Informático Industrial	60	h	30,00 €	1.800,00 €
Total Ejecución Material				2.005,60 €
13 % Gastos generales				260,73 €
6 % Beneficio industrial				120,34 €
Importe de Ejecución				2.386,67 €
21 % I.V.A.				501,20 €
Importe de Contrata				2.887,87 €

Tabla 6.2: Presupuesto de prototipo de sensor inteligente de CO2

Ascendiendo el presupuesto general a la expresada cantidad de **DOS MIL OCHOCIENTOS OCHENTA Y SIETE EUROS CON OCHENTA Y SIETE CÉNTIMOS (2.887,87€)**.