Escuela Politécnica Superior de Jaén
Máster Universitario en Ingeniería Mecatrónica

UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior (Jaén)

Trabajo Fin de Máster

PROTOTIPO DE UN SISTEMA DE MONITORIZACIÓN DE LA CALIDAD DEL AIRE EN INTERIORES

Alumno/a: Merino Arance, Pedro José

Tutor/a: Macarena Espinilla Estévez

Cotutor/a: Alicia Montoro Lendínez

Dpto.: Departamento de Informática

Septiembre, 2023

Universidad de Jaén
Departamento de Informática

Doña Macarena Espinilla Estévez y doña Alicia Montoro Lendínez, tutores del Trabajo Fin de Máster titulado: "**Prototipo de sistema de monitorización de la calidad del aire en interiores**", que presenta Pedro José Merino Arance, autoriza su presentación para defensa y evaluación en la Escuela Politécnica Superior de Jaén.

Jaén, julio de 2023

El alumno: Los tutores:

Pedro José Merino Arance

Macarena Espinilla Estévez
Alicia Montoro Lendínez

Agradecimientos

Me gustaría mostrar mi agradecimiento a mis tutores, Macarena y Alicia. Su positividad y paciencia han ayudado mucho a la realización de este trabajo. También dar las gracias a mis padres y mi hermano por el apoyo recibido diariamente desde que empecé esta aventura, siempre presentes en los momentos más difíciles.

FICHA DEL TRABAJO FIN DE TÍTULO	
Titulación	Master universitario en Ingeniería Mecatrónica
Modalidad	Proyecto de Ingeniería
Especialidad (solo TFG)	Elija un elemento.
Mención (solo TFG)	Elija un elemento.
Idioma	Español
Tipo	Elija un elemento.
TFT en equipo	No
Fecha de asignación	28/09/2022
Descripción corta	Los ambientes inteligentes son entornos donde parte de sus objetos u ocupantes se encuentran envueltos o asistidos por una red de sensores y actuadores con el objetivo de buscar el bienestar de los ocupantes y conseguir una relación más amigable, sostenible y segura del ocupante en dicho entorno. El presente trabajo fin de máster está centrado en el análisis, diseño y desarrollo de un sistema de monitorización de la calidad del aire en espacios interiores. Para ello, se utilizarán dispositivos de bajo coste, como las placas de desarrollo NodeMCU, para orquestar el sistema, sensores de gases para medir la calidad del aire y sensores de inerciales para conocer el estado de las puertas y ventanas del entorno interior. Los datos extraídos de los sensores serán enviados y procesados en una arquitectura con conexión a una plataforma IoT de terceros para su visualización. Finalmente, se llevarán a cabo diferentes escenarios para evaluar el sistema desarrollado.

NORMAS APLICADAS EN ESTE DOCUMENTO	
LOCALES	
TFT-UJA:2017	Normativa de Trabajos Fin de Grado, Fin de Máster y otros Trabajos Fin de Título de la Universidad de Jaén (Normativa marco UJA aprobada en Consejo de Gobierno)
TFT-EPSJ:2017	Normativa sobre Trabajos Fin de Grado y Fin de Máster en la Escuela Politécnica Superior de Jaén (Normativa EPSJ aprobada en Junta de Escuela)
TFT-EPSJ	Criterios de evaluación y normas de estilo para TFG y TFM de la Escuela Politécnica Superior de Jaén
NACIONALES	
UNE 157001:2014	Criterios generales para la elaboración formal de los documentos que constituyen un proyecto técnico
UNE 157801:2007	Criterios generales para la elaboración de proyectos de sistemas de información
INTERNACIONALES	
ISO 2145:1978	Documentación - Numeración de divisiones y subdivisiones en documentos escritos
APA 6ª edición	Estilo de referencias y citas de APA (American Psychological Association)

CONTENIDO

INDICE DE FIGURAS

INDICE DE TABLAS

1. Introducción.

En este trabajo de fin de máster (TFM) se presenta el prototipo de un sistema de monitorización de la calidad del aire en interiores. En este primer capítulo, se presentará el marco teórico del proyecto desarrollado, su justificación, una idea general sobre qué consiste y las motivaciones que han llevado a escogerlo.

1.1 Marco teórico.

La polución, la contaminación, el cambio climático y otros fenómenos están generando perturbaciones ambientales de imprevisibles consecuencias en la salud de las personas y, en general, del equilibrio de la vida y el ecosistema. En España, en 2020, se desarrolló un Índice Nacional de Calidad del Aire (ICA) que permite a todos los usuarios comprobar la calidad del aire en todas las ciudades y regiones españolas. El ICA utiliza datos en tiempos real procedentes de las estaciones de medición de la calidad del aire, comunicados cada hora por las redes de calidad del aire que operan en el territorio nacional [1]. Sin embargo, en espacios cerrados, como universidades, residencias, oficinas, colegios, entre otros, donde la concentración de contaminantes puede ser aún mayor que en el exterior, no se lleva con exhaustividad un control de la calidad del aire.

La mayoría de nuestras actividades transcurren en ambientes cerrados, ya sea doméstico o laboral. En estos espacios es esencial una renovación del aire adecuada para mejorar la calidad del aire. La renovación del aire se mide con la ACH (‘Air Changes per Hour’). La determinación de la ACH para un espacio determinado se determina mediante varios métodos. Algunos se basan en la medida de caudales de entrada y salida y otras se basan en la medida de la concentración de dióxido de carbono (CO_2). La concentración de CO_2 es un buen indicador de la tasa de renovación de aire de un espacio. La concentración de CO_2 en un espacio cerrado va a depender del volumen de la sala, el número de ocupantes, su edad y la actividad realizada.

Un estudio realizado en 2012 [2], en la universidad Upstate Medical, en Nueva York, demostró que la exposición a grandes concentraciones de CO_2 pueden ser perjudicial para la salud de las personas. En particular, este estudio hace referencia a una reducción significativa del rendimiento personal, por ejemplo, en la toma de decisiones, además de afectar el estado físico (dolores de cabeza, mareos…) o fisiológico (aumento de la presión sanguínea).

Los seres humanos exhalamos CO_2 al respirar, por lo que, los niveles de CO_2 aumenta con el paso del tiempo en habitáculos cerrados con mala ventilación. Dada esta circunstancia, se hace indispensable ventilar la habitación. Si la ventilación es natural se recomienda ventilación cruzada (apertura de puertas y/o ventanas opuestas o al menos lados diferentes de la sala).

Al margen de esta problemática, la ventilación de espacios cerrados también ayuda a reducir la transmisión de enfermedades virales que se propagan mediante los aerosoles que producimos las personas al hablar, cantar, gritar…Durante el pasado siglo, se pensaba que la transmisión de virus respiratorios se producía mayoritariamente mediante las gotas (tamaño >100μm), generadas al toser, o a través de superficies infectadas (fómites). Durante la pandemia de SARS-CoV-2 (COVID-19), investigadores proporcionaron numerosas pruebas que evidencian el predominio de la transmisión del COVID-19 en habitáculos cerrados y mal ventilados. Esto implica solamente a los aerosoles (partículas menores de 100 μm) ya que a las gotas o los fómites no les afecta la ventilación [3]. Tal es la importancia de esto, que, desde las grandes instituciones, como el Gobierno de España, se puso a disposición de la ciudadanía información relativa a concentración de partículas, sistema de ventilación o distancias de seguridad en espacios cerrados [4].

Con los puntos mencionados, se hace necesario la utilización de nuevas tecnologías para la obtención de ciertos datos, que a posteriori, se analizaran con el objetivo de aprovechar esa información para mejorar nuestra calidad de vida.

Por tanto, este TFM se ha centrado en abordar el problema de la calidad de aire en ambientes cerrados mediante su monitorización haciendo uso del paradigma IoT, que nos proporciona un gran despliegue de sensores, dispositivos y actuadores conectados entre sí capaces de intercambiar datos.

1.2 Propósitos generales y objetivos.

1.2.1 Objetivos generales.

Este trabajo de fin de master tiene como objetivo la monitorización de la concentración de CO_2 en un espacio cerrado con el fin de obtener el estado de la calidad del aire en cada momento. Se puede obtener la información del estado de la calidad del aire tomando en cuenta diversas variables. En este TFM nos centraremos en el grado de apertura de una puerta, el grado de apertura de una ventana y de la actividad física que realiza la persona que habita en ese espacio cerrado. El sistema

IoT está compuesto por diferentes dispositivos conectados entre sí que tendrá las siguientes características:

- Integración de un acelerómetro en la puerta de la habitación con el fin de obtener el grado de apertura de esta.
- Integración de un acelerómetro en la ventana de la habitación con el objetivo de obtener el grado de apertura de esta.
- Integración de un sensor de gas, localizado en una zona intermedia entre la ventana y la puerta.
- Recepción en tiempo real de los datos recolectados que se envían de forma inalámbrica a un sumidero central de datos. Este sumidero tiene la misión tanto de almacenar los datos provenientes de los diferentes sensores cómo la de enviar toda esta información a una plataforma IoT, donde se visualizarán todos estos datos.

1.2.2 Objetivos específicos.

Los objetivos específicos de este TFM, que nos ayudarán a cumplir los objetivos generales, son:

- Revisar la bibliografía de las tecnologías asociadas en este trabajo.
- Definir los distintos procesos necesarios para desarrollar el sistema de monitorización.
- Diseñar un sistema IoT mediante el despliegue de los diferentes dispositivos que permitirán la monitorización del espacio cerrado.
- Diseñar e implementar una arquitectura capaz de recolectar los datos provenientes de los diferentes sensores, almacenarlos en un servidor cloud y visualizarlos en una plataforma IoT.
- Validar el sistema IoT mediante diferentes pruebas.
- Realizar los manuales asociados.
- Redactar una memoria que recoja todo el trabajo desarrollado.

1.3 Estado del arte.

En este apartado se realiza una revisión de la bibliografía con el objetivo de analizar las diferentes soluciones que han ido surgiendo, a lo largo del tiempo, para la monitorización de gases a partir de dispositivos IoT.

Antes de comenzar nuestra investigación, debemos tener claro el concepto de "Internet de las Cosas", o IoT. Este concepto apareció por primera vez en un discurso de Peter T.Lewis, en el 15º fin de semana legislativo anual de la fundación del Caucus Negro del congreso en Washington DC, publicado en 1985 [5]. Peter definió "Internet de las Cosas" como "una integración de personas, procesos y tecnología con dispositivos y sensores conectables para permitir el monitoreo remoto, el estado, la manipulación y la evaluación de las tendencias de dichos dispositivos". De este término ha surgido varias definiciones a lo largo de estos últimos años debido al crecimiento exponencial del IoT, que pueden llevar a confusión. Por ejemplo, Oxford Dictionaries [6] ofrece una definición que invoca a Internet como un elemento de IoT:

"Interconexión a través de Internet de dispositivos de computación integrados en objetos cotidianos, que les permite enviar y recibir datos".

En cambio, la recomendación ITU-T Y.2060 que publicó la Unión Internacional de Telecomunicaciones (UIT) en 2012 [7], "Overview of the Internet of things", discute el concepto de interconectividad, pero no vincula IoT específicamente con Internet:

"Infraestructura mundial para la sociedad de la información que propicia la prestación de servicios avanzados mediante la interconexión de objetos (físicos y virtuales) gracias a la interoperatividad de tecnologías de la información y la comunicación presentes y futuras".

Para los propósitos de este trabajo, los términos "Internet de las Cosas" o "IoT" se referirá a cosas conectadas a la red. Estas "Cosas" deben tener tecnología incorporada que le permita conectividad entre ellas y/o Internet, además de generar (o no) datos para poder transmitirlos y poder realizar alguna acción (o no) en función de los datos que recibe.

La integración del paradigma IoT dentro de monitorización de los gases en espacios cerrados es muy importante para el éxito. Sin embargo, no todos los gases son importantes para el estudio de la calidad del aire en interiores. Algunos de los gases más relevantes para su estudio, según la Organización mundial de la Salud [8], son los siguientes:

- Dióxido de carbono (CO_2): Es un gas importante de efecto invernadero. Se produce naturalmente en la respiración de los seres vivos. Grandes concentraciones de este gas reducen significativamente el desempeño en las tomas de decisiones. Se estima que una habitación bien ventilada debe tener una concentración de CO_2 de 800-900 ppm [4].

- Monóxido de carbono (CO): Resultado de una combustión incompleta. A niveles bajos, la exposición a este gas puede causar fatiga. Las personas con problemas cardíacos pueden experimentar dolor en el pecho. A altos niveles, el CO desencadena dolores, mareos, confusión y/o náuseas. También puede matarte.
- Benceno (C_6H_6): Es un carcinógeno genotóxico en humanos. Se encuentra tanto en ambientes cerrados como en ambientes cerrados, aunque los niveles de este gas suelen ser más altos en espacios cerrados. Este gas se produce al fumar, al utilizar disolventes para la limpieza o en el uso de materiales de construcción que liberan este gas.
- Formaldehido (CH_2O): A niveles bajos, puede causar irritación en los ojos en humanos. Una exposición prolongada a este gas puede causar cáncer a largo plazo.
- Dióxido de nitrógeno (NO_2): A ciertos niveles de este gas, los asmáticos exhiben pequeñas disminuciones de la función pulmonar.
- Naftalina ($C_{10}H_8$): Sólido blanco que se produce, por ejemplo, cuando se quema tabaco. Los principales problemas de salud por la exposición de naftalina son lesiones en las vías respiratorias, incluido anemia hemolítica en humanos.
- Radón (Rn): Es clasificado por el Centro Internacional de Investigadores sobre el cáncer como un carcinógeno humano.
- Hidrocarburos aromáticos policíclicos (PAHs): Son potentes carcinógenos. Se producen durante la combustión incompleta de, por ejemplo, tabaco.
- Tricloroetileno (TCE): Las personas puede estar expuestas al utilizar adhesivos, lubricantes, barnices o determinados productos de la limpieza. A altos niveles, puede causar daño neurológico, que afecta especialmente a los nervios ópticos y nervio trigémino.
- Tetracloroetileno (C_2Cl_4): En humanos, la exposición a este elemento puede afectar al sistema nervioso central, a los ojos, a la piel, a los riñones, a los pulmones o a las membranas mucosas.

Centrándonos en la monitorización de la calidad del aire en espacios cerrados, a lo largo de los años se ha intentado desarrollar diferentes sistemas de bajo costo, utilizando (o no) “Internet de las Cosas”.

En 2004, diferentes investigadores desarrollaron una nariz electrónica de bajo costo [10]. Mediante el uso de una combinación de técnicas avanzadas de reconocimiento de patrones y una matriz optimizada de sensores de gas, los investigadores se centraron en la cuantificación de óxidos de nitrógeno, monóxido de carbono, junto con los compuestos orgánicos volátiles (COVs) y humedad relativa. Los resultados que obtuvieron mostraron la viabilidad de estas narices electrónicas para detectar CO y NO_2 en concentraciones inferiores a los valores umbral de la calidad de aire en interiores.

En 2014, Kim-Yoon, Chao-Hsien y Sang-Moon [11] se centraron en siete gases (CO_2, COVs, SO_2, NOx, CO, PM y ozono) para comprobar la calidad del aire en interiores. Los experimentos se llevaron a cabo en diferentes localizaciones: una iglesia grande, un aula de tamaño mediano y una sala de estar de tamaño pequeño con el objetivo de probar diferentes factores en el impacto de la calidad del aire en interiores. Algunos de esos factores son el viento, la localización, la densidad de personas y el tamaño de la habitación. Se dieron cuenta que los sensores de gases consumían mucha energía, por lo que era necesario pensar muy bien en la selección adecuada de los nodos sensores.

Un año más tarde, en 2015, Yu y Lin [12] construyeron un sistema de control inteligente e inalámbrico con el objetivo de abordar los problemas de salud causados por la calidad del aire en interiores. El sistema se compone de tres partes diferentes:
1) Adquisición de datos que ayudan a obtener valores sobre indicadores ambientales como la concentración de CO_2, la humedad relativa y temperatura a través de mecanismos de sondeo.
2) Análisis de datos, responsable de recopilar datos y analizar las tendencias de la calidad de la calidad del aire en las instalaciones.
3) Retroalimentación de datos para desencadenar acciones necesarias basadas en resultados parcias. Puede enviar un mensaje de advertencia o controlar la velocidad del ventilador automáticamente.

Utilizaron ZigBee como protocolo de comunicación. Este sistema se implementó, con éxito, en nueve localizaciones de Taiwán, entre las que destacan, la Administración de Protección Ambiental, universidades o escuelas primarias.

En los años recientes, se ha seguido buscando dar un paso más en el desarrollo de este tipo de sistemas. Por ejemplo, en 2017, investigadores de Korea [13] desarrollaron un microchip utilizando 6 sensores atmosféricos: COVs, cantidad de luz, humedad temperatura, polvo fino y CO_2. Además, utilizaron modelos de "Deep learning" para estimar los cambios atmosféricos. Más reciente aún, en 2019, en la Universidad de Extremadura [14] se desarrolló un sistema de red de sensores inalámbricos distribuida conectada a un sistema de computación en la nube. Este sistema en la nube permite almacenar, monitorear, procesar y visualizar los datos recibidos de la red de nodos.

Otros investigadores han intentado desarrollar sistemas para monitorizar la calidad del aire en espacio cerrados alejados de la arquitectura IoT, revelando un bajo rendimiento comparados con los sistemas desarrollados con la tecnología IoT. La desventaja más significativa de la plataforma C-air presentada por Wu et al. [15] fue que este estudio estaba limitado a los niveles PM y ese no es el único factor que afecta a la calidad del aire en interiores. Moreno-Rangel et al. [16] presentó un estudio fiable con monitores FOOBOT considerando varios parámetros para la medición de la calidad del aire en interiores, pero, en este caso, el problema fue el desafío que suponía calibrar el sensor para asegurar el rendimiento deseado.

Debido a esto, se hace indispensable la utilización de la arquitectura IoT para el desarrollo de nuestro trabajo. Además, se hace visible que la inmensa mayoría de los sistemas desarrollados se han elaborado en un entorno de laboratorio, donde los recursos y herramientas a la disposición de los investigadores son casi ilimitados. En nuestro caso, el sistema se desarrollará en un entorno doméstico con recursos más limitados a nuestra disposición.

1.4 Análisis.

En este apartado se presenta el análisis llevado a cabo para el desarrollo del sistema de monitorización de la calidad de aire en interiores. Para ello, es muy importante la determinación de los requisitos funcionales y no funcionales.

1.4.1 Requisitos: funcionales y no funcionales.

Los requisitos funcionales definen una función del sistema software o sus componentes. Una función es descrita como un conjunto de entradas, comportamientos y salidas. Los requisitos funcionales pueden ser: cálculos, detalles técnicos, manipulación de datos y otras funcionalidades específicas que el sistema

debe cumplir [17]. Ayudándonos de esta definición, los requisitos funcionales de nuestro proyecto son los siguientes:

- El sistema debe tener la capacidad de obtener el nivel de concentración de los gases que se encuentran en la habitación.
- El sistema debe detectar las variaciones en los niveles de concentración debido a la apertura o cierre tanto de la ventana como de la puerta de la habitación.
- El sistema debe tener la capacidad de obtener los grados de apertura y cierre tanto de la ventana como de la puerta de la habitación.
- Los sensores se comunicarán con el nodo sumidero de forma inalámbrica.
- Las mediciones recolectadas por los diferentes sensores deben ser persistentes en una base de datos.
- Los datos recogidos por los sensores deben ser monitoreados en tiempo real en una plataforma IoT.

Los requisitos no funcionales explican las limitaciones y restricciones del sistema a diseñar. Estos requisitos no tienen ningún impacto en la funcionalidad del sistema. Los requisitos no funcionales de nuestro sistema son los siguientes:

- Realizar un precalentamiento superior a las 24 horas del sensor SEN0159 en su primera utilización. Necesitan este precalentamiento para su correcta calibración.
- El sensor SEN0159 se localizará en una posición intermedia entre la ventana y la puerta de la habitación para obtener unos datos mejor calibrados respecto a la ventilación obtenida tanto por la puerta como la ventana.
- Tanto la puerta como la ventana estarán cerradas la primera vez que ejecutemos el código programado para la utilización de los acelerómetros. De esta manera, los acelerómetros reconocerán esa posición como la posición inicial y calcularán el ángulo de apertura y cierre a partir de esa posición.
- La plataforma IoT que se ha seleccionado para la visualización de los datos estará alojada en una red local debido a que esta TFM se ha desarrollado en la Universidad de Jaén y dicha universidad no permite actualmente conectar los dispositivos con plataformas IoT alojadas en la nube por motivos de seguridad.

1.5 Planificación de las tareas.

En este apartado se describe la planificación de las tareas que se han llevado a cabo en este TFM.

La siguiente tabla muestra una pequeña descripción de todas las tareas llevadas a cabo y una estimación del tiempo invertido en ellas:

TAREAS	TIEMPO (h)
Revisión de literatura sobre la calidad del aire en interiores.	**15**
Revisión de literatura sobre IoT.	**15**
Revisión de literatura sobre la monitorización de la calidad del aire en interiores.	**25**
Análisis de los requisitos del sistema IoT.	**6**
Puesta en marcha de los sensores.	**69**
Detección de CO_2.	51.5
Búsqueda de información, documentación y ejemplos sobre el sensor SEN0159.	1
Realización del montaje del sensor SEN0159 y la placa ESP32 en la protoboard.	0.5
Desarrollo de script para la detección del CO2.	1
Calibración del sensor de gas SEN0159.	48
Primeras pruebas del funcionamiento del sensor SENS0159.	1
Ángulo de apertura de la ventana.	14.5
Búsqueda de información, documentación y ejemplos sobre el sensor MPU6050.	5
Realización del montaje del sensor MPU6050 y la placa ESP8266 en la protoboard.	0.5
Desarrollo del script para cálculo del ángulo de apertura de la ventana.	5
Calibración del sensor MPU6050.	2
Montaje de la protoboard en la ventana y primeras pruebas.	2
Ángulo de apertura de la puerta.	3
Realización del montaje del sensor MPU6050 y la placa ESP8266 en la protoboard.	0.5
Desarrollo de script para cálculo del ángulo de apertura de la puerta.	0.5
Calibración del sensor MPU6050.	1
Montaje de la protoboard en la ventana y primeras pruebas.	1
Inclusión del servicio de mensajería MQTT en los sensores.	**8.75**
Puesta en punto de la Raspberry Pi.	2.75
Búsqueda de información sobre como instalar el sistema operativo en la Raspberry Pi.	0.5
Instalación del sistema operativo Raspberry PI OS lite (64 bits)	0.25
Activar el servicio SSH.	0.25
Conectar la Raspberry Pi al WIFI.	0.25
Búsqueda de información sobre mosquitto e instalación de este en la Raspberry Pi.	1.5
Detección de CO_2.	4
Búsqueda de información sobre MQTT y su implementación en Arduino.	2.5
Desarrollo del script detección de CO2 incorporando el protocolo MQTT.	1
Primeras pruebas del sensor SEN0159 publicando un topic que recoge la Raspberry Pi.	0.5
Ángulo de apertura de la ventana.	1
Desarrollo del script ángulo de apertura de la ventana incorporando el protocolo MQTT.	0.5
Primeras pruebas del sensor MPU6050 publicando un topic que recoge la Raspberry Pi.	1
Ángulo de apertura de la puerta.	1
Desarrollo del script ángulo de apertura de la puerta incorporando el protocolo MQTT.	0.5

Primeras pruebas del sensor MPU6050 publicando un topic que recoge la Raspberry Pi.	0.5
Diseño de la base de datos del sistema IoT.	**10.25**
Búsqueda de información sobre MongoDB.	4
Instalación de PyCharm para el desarrollo de aplicación en Python.	0.25
Desarrollo de script para el almacenamiento de los datos en MongoDB.	5
Pruebas con los tres sensores funcionando.	1
Inclusión de una plataforma IoT para la visualización de los datos en tiempo real.	**9.5**
Investigación sobre las diferentes alternativas para la elección de la plataforma.	3
Búsqueda de información sobre Home Assistant.	3
Puesta en punto de la Raspberry Pi.	2
Instalación de Home Assistant Supervisor.	1.5
Instalación de mosquitto y el complemento file editor en Home Assistant.	0.5
Modificación de los scripts para la inclusión de la nueva IP del servidor MQTT.	0.5
Diseño de la dashboard en Home Assistant para la visualización de los datos.	1
Primeras pruebas con los tres sensores funcionando.	1
Diseño de los soportes en Catia V5.	**3**
Impresión 3D de los soportes.	**12**
Desarrollo de las pruebas del sistema IoT.	**15**
Corrección de errores.	**15**
Redacción de memoria.	**80**
Total	**284**

Tabla 1. Tareas realizadas y tiempo invertido.

1.5.1 Diagrama de Gantt.

El diagrama de Gantt es una metodología de representación de actividades o tareas que pretende dar una visión generalizada sobre el tiempo dedicado a cada actividad contemplada de forma independiente dentro de un proceso. En otras palabras, el diagrama de Gantt es un tipo de gráfico que representa las actividades de forma independiente con el objetivo de tener una imagen general de cómo evolucionan las tareas a través del tiempo [18].

En la siguiente ilustración se muestra el diagrama de Gantt de nuestro proyecto:

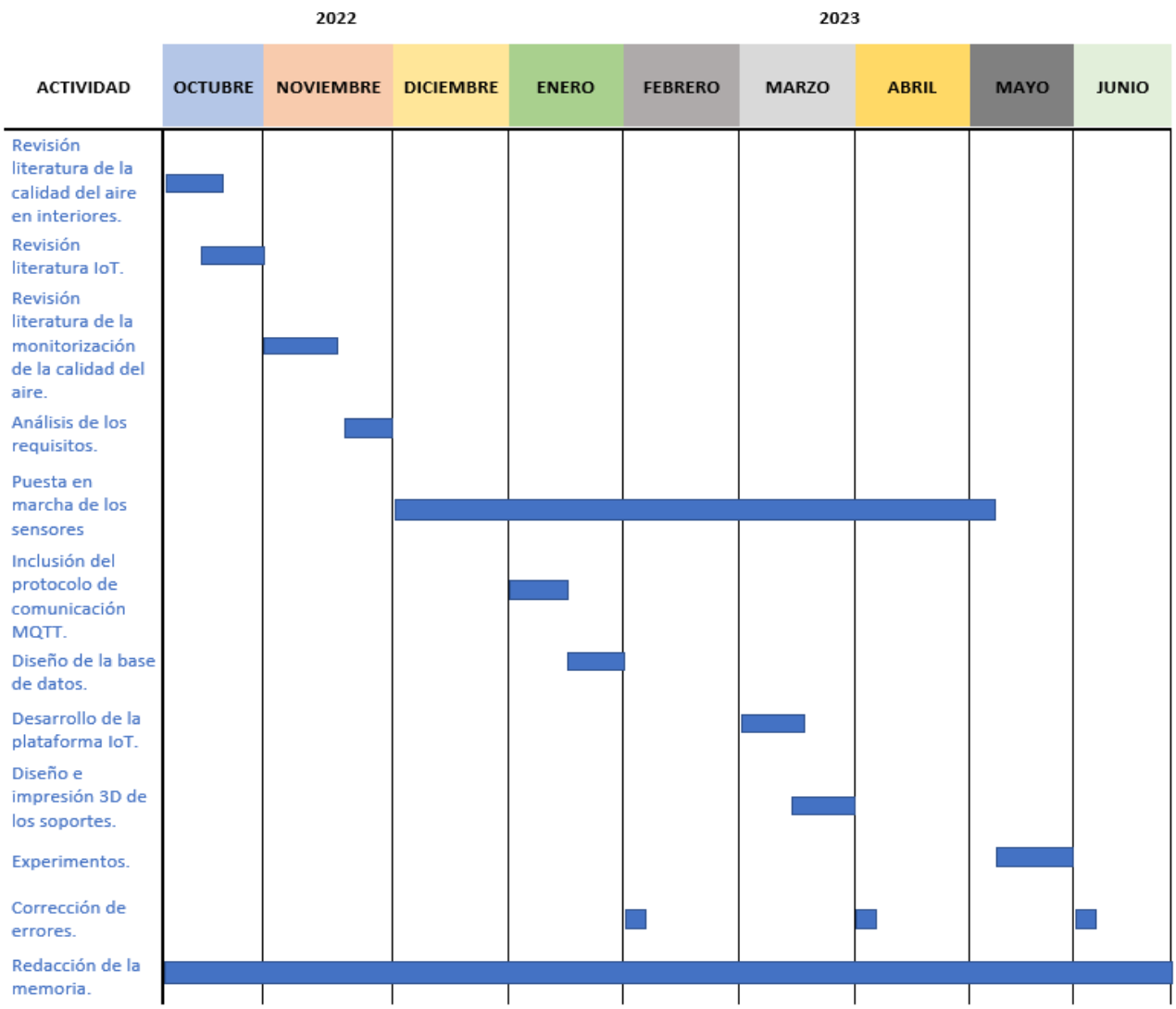


Figura 1. Diagrama de Gantt del proyecto.

En este diagrama de Gantt se puede observar que el proyecto se empezó con una revisión literaria para informarnos sobre todos los aspectos importantes que hay que tener en cuenta para saber la calidad del aire en interiores. Además, también se hizo una revisión literaria del concepto IoT, imprescindible tenerlo claro para este proyecto, y las metodologías más importantes desarrolladas a lo largo del tiempo para la monitorización de la calidad del aire en interiores. A partir de diciembre se empezó con el desarrollo de toda nuestra arquitectura IoT en el que se incluye la puesta en marcha de los sensores, la comunicación con el nodo sumidero, el diseño de la base de datos y el desarrollo de la plataforma IoT. Para finalizar se hizo las diferentes pruebas al sistema IoT desarrollado para evaluar su funcionamiento. La redacción de la memoria se ha realizado durante los 9 meses que ha durado el proyecto.

1.6 Estimación de los costes.

En este apartado se va a realizar una estimación de los costes totales para el desarrollo y ejecución del proyecto. Se tendrá en cuenta tanto costes directos como costes indirectos.

Para el cálculo del presupuesto se ha considerado que el desarrollo de este sistema es parte de un proyecto aún mayor de IoT dentro de una empresa. El proyecto tiene asociado un jefe de proyectos, un analista programador junior y un programador junior.

1.6.1 Costes directos.

Los costes directos son aquellos costes que tienen una implicación directa con el servicio que ofrecemos o con aquello que producimos. En nuestro caso:

- Mano de obra directa (jefe de proyectos, analista programador y programador).
- Coste hardware.
- Coste software.

La mano de obra directa se refiere al personal que participa directamente en la fabricación de un producto específico o en la realización de un servicio. Para calcular el coste asociado se debe tener en cuenta el tiempo empleado por cada trabajador y la tarea realizada por este, para determinar los sueldos. Para el cálculo de los costes del personal se utilizarán los datos proporcionados en la siguiente tabla:

Conceptos	Jefe de proyecto	Analista programador	Programador
Situación personal y familiar	Soltero y sin hijos	Soltero y sin hijos	Soltero y sin hijos
Discapacidad	No	No	No
Salario bruto anual (€)	40000	18896	18500
Pagas extras prorrateadas	2	2	2
Tipo de contrato	Temporal	Temporal	Temporal
Duración del contrato	6 meses	6 meses	6 meses

Tabla 2. Datos necesarios para la obtención de los costes de la mano de obra directa.

A partir de estos datos se puede calcular el coste para la empresa por la contratación del personal durante los 6 meses:

- **Jefe de proyectos**
  - Salario bruto mensual: 40000/14 → 2857.14 €

- Paga extra: (2857.14*2) /12 → 476.19 €
- Base de cotización: 2857.14+476.19 → 3333.3 €
- Gastos de la seguridad social → 1083.33 €
  - Contingencias comunes (23.6%): 3333.3*0.236 → 786.67 €
  - AT y EP (1.5%): 3333.3*0.015 → 50 €
  - Desempleo (6.7%): 3333.3*0.067 → 223.33 €
  - Formación profesional (0.6%): 3333.3*0.006 → 20 €
  - FOGASA (0.1%): 3333.3*0.001 → 3.33 €
- Coste total mensual: 3333.3+1083.33 → 4416.67 €

- **Analista programador junior**
  - Salario bruto mensual: 18896/14 → 1349.71 €
  - Paga extra: (1349.71*2) /12 → 224.95 €
  - Base de cotización: 1349.71+224.95 → 1574.67 €
  - Gastos de la seguridad social → 511.76 €
    - Contingencias comunes (23.6%): 1574.67*0.236 → 371.62 €
    - AT y EP (1.5%): 1574.67*0.015 → 23.62 €
    - Desempleo (6.7%): 1574.67*0.067 → 105.5 €
    - Formación profesional (0.6%): 1574.67*0.006 → 9.45 €
    - FOGASA (0.1%): 1574.67*0.001 → 1.57 €
  - Coste total mensual: 1349.71+511.76 → 2086.43 €
- **Programador junior**
  - Salario bruto mensual: 18500/14 → 1321.43 €
  - Paga extra: (1321.43*2) /12 → 220.23 €
  - ase de cotización: 1321.43+220.23 → 1541.67 €
  - Gastos de la seguridad social → 501.04 €
    - Contingencias comunes (23.6%): 1541.67*0.236 → 363.83 €
    - AT y EP (1.5%): 1541.67*0.015 → 23.125 €
    - Desempleo (6.7%): 1541.67*0.067 → 103.3 €
    - Formación profesional (0.6%): 1541.67*0.006 → 9.25 €
    - FOGASA (0.1%): 1541.67*0.001 → 1.54 €
  - Coste total mensual: 1541.67+501.04 → 2042.73 €

Concepto	Jefe de proyecto	Analista programador	Programador
Salario bruto mensual (€)	2857,142857	1349,714286	1321,428571
Base de cotización (€)	3333,333333	1574,666667	1541,666667
Gastos de seguridad social (€)	1083,333333	511,7666667	501,0416667
Coste total mensual (€)	4416,666667	2086,433333	2042,708333
Tiempo	6 meses	6 meses	6 meses
Coste total (€)	**26500**	**12518,6**	**12256,25**

Tabla 3. Costes directos debido a la mano de obra.

Además de la mano de obra, se tiene que calcular los costes debido tanto al hardware como al software utilizado. En la siguiente tabla se muestra los costes debido al hardware utilizado:

Hardware	Cantidad	Precio unitario (€)	Precio total (€)
ESP8266	2	8,47	16,94
ESP32	1	12	12
SEN0159	1	46,39	46,39
MPU6059	2	6,79	13,58
Placaboard	3	3,33	9,99
Raspberry Pi Model 3	1	31,81	31,81
Cables	40	0,1	4
Resistencias	3	0,1	0,3
Powerbank	2	28,87	57,74
MB102 Power supply module	1	5,99	5,99
Cable alimentación 5V/2A	4	9,99	39,96
		Total (€)	**238.7**

Tabla 4. Costes directos debido al hardware.

El software utilizado es de libre distribución por lo que no se ha generado ningún coste. En la siguiente tabla se muestra los costes debido al software utilizado:

Software	Cantidad	Precio unitario (€)
IDE Arduino	1	0
Librería sensor MPU	1	0
Librería MQTT	1	0
Catia V5 licencia estudiante	1	0
PyCharm Community	1	0
MongoDB	1	0
Home Assistant	1	0
Python y librerías de Python	1	0
	Total (€)	**0**

Tabla 5. Costes directos debido al software.

1.6.2 Costes indirectos.

Los costes indirectos son aquellos costes que no se pueden imputar directamente al servicio que ofrecemos o al producto que fabricamos. En nuestro caso:

- Internet.
- Gastos de contingencias (Luz, agua…).
- Material fungible.

En la siguiente tabla se muestra los costes indirectos derivados por los servicios necesarios para el desarrollo de nuestro proyecto:

Servicio	€/mes	Tiempo (meses)	Coste total €
Internet	80	6	480
Gastos de contingencia	100	6	600
Material fungible	70	1	70
		Total (€)	**1150**

Tabla 6. Costes indirectos.

1.6.3 Costes totales.

Para el cálculo de los costes totales se ha estimado un beneficio de un 6%.

Tipo de coste	Coste total €
Mano de obra (Jefe de proyecto)	26500
Mano de obra (Analista programador)	12518,6
Mano de obra (Programador)	12256,25
Costes hardware	238.7
Costes software	0
Costes indirectos	1150
Total costes	52663,55
Beneficio (6%)	3159,813
Total	**55823,363**

Tabla 7. Costes totales.

1.7 Metodología.

En este apartado se va explicar las diferentes metodologías existentes a la hora de afrontar el desarrollo de un software. Cada metodología de desarrollo de software está diseñada para un tipo de proyecto en específico. Para ello se toman diferentes factores cómo las herramientas y dispositivos con los que se trabajarán, la organización de las tareas o el equipo de trabajo con el que se cuenta. Además, se seleccionará la metodología utilizada en este TFM.

1.7.1 Metodologías tradicionales.

Las metodologías tradicionales [19], también conocidas como modelos de proceso prescriptivo, se crearon con el propósito de mejorar el pésimo tratamiento que se le daba al software durante sus primeros años de vida. Se rigen bajo una estructura secuencial inalterable basada en etapas. Dichas etapas son:

- Análisis de requerimiento.

- Diseño.
- Programación.
- Pruebas.
- Mantenimiento.

Se les da una gran importancia a los requerimientos del proyecto dado que se precisa un amplio estudio para su definición. Una vez establecidos estos requerimientos no pueden alterarse de ninguna manera.

Las principales metodologías tradicionales son las siguientes:

- **Espiral [20].**

El modelo de desarrollo en espiral es un modelo evolutivo de cuatro fases:

- Planificación.
- Análisis de riesgos.
- Desarrollo.
- Evaluación.

A lo largo de la aplicación del modelo de desarrollo en espiral, estas cuatro fases se van a repetir, con la diferencia de que el proyecto irá aumentando su complejidad, lapsos de tiempo de ejecución, volumen de tareas…

Es un modelo que va creciendo, cuidando los recursos y siendo cautelosos de los riesgos. Esta modalidad de gestión no asumirá el riesgo más importante, hasta no concluir un ciclo de 4 etapas inicial.

Este modelo evolutivo, escalable y espiral, conforme avancen los ciclos, irá asumiendo mayores riesgos, ya que tiene la tranquilidad de haber desarrollado y evaluado el producto final, pero a menor escala. Tiene la capacidad de evolucionar su complejidad con cada ciclo.

- **Cascada [21].**

El modelo de cascada es un método de gestión de proyectos, en el que el proyecto se divide en distintas fases secuenciales, donde el equipo puede pasar a la siguiente fase sólo cuando se haya completado la fase anterior. Podemos dividir el modelo en las siguientes fases:

- Requisitos: Es la fase más importante. Durante esta fase normalmente se realizan entrevistas, reuniones e intercambios de opiniones para definir los requisitos para el proceso de desarrollo y el resultado final del proyecto.
- Diseño y construcción: Esta etapa puede contener procesos de implementación, desarrollo y codificación.

- Fase de prueba: En esta etapa, los especialistas responsables prueban el software (u otro producto que se desarrolla en el proyecto) y detectan errores.
- Instalación: Es una fase en la que el producto sale par el uso de acuerdo con todos los requisitos.
- Mantenimiento: El producto final se entrega al cliente. Para algunos proyectos, por ejemplo, un software, se necesita el mantenimiento continuo.

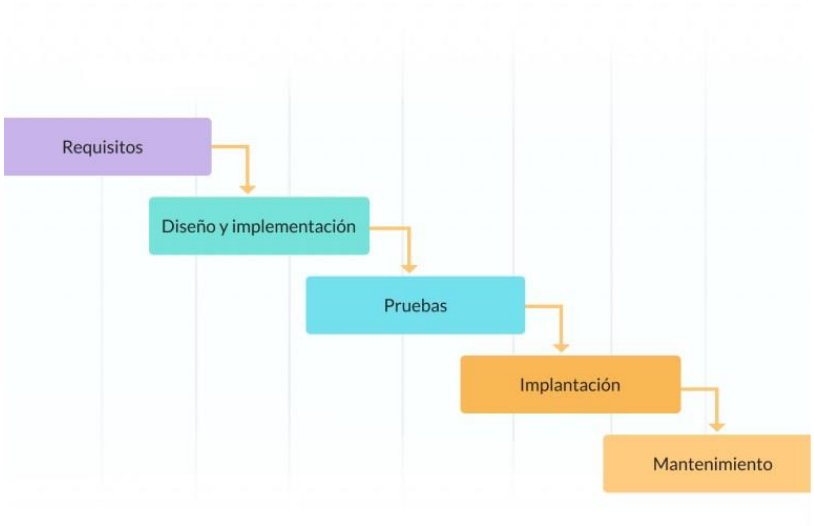


Figura 2. Modelo en cascada.

Este método se aplica:

- Cuando hay una visión clara de lo que debería ser el producto final.
- Cuando los clientes no tienen posibilidad de cambiar el alcance del proyecto una vez que ha comenzado.
- Cuando el concepto y la definición son las claves del éxito (pero no la velocidad).
- Cuando no hay requisitos ambiguos.

- **Prototipo.**

Destaca por no precisar de requerimientos robustos o estables para iniciar el proyecto. Facilita la presentación de los resultados del proyecto a los clientes. Promueve un mayor acercamiento directo a las necesidades del cliente.

El prototipo generado suele confundirse con el producto final, aun cuando no está finalizado. Se precisa de la participación del cliente en el proceso.

- **Incremental [22].**

El modelo incremental de gestión de proyectos tiene como objetivo un crecimiento progresivo de la funcionalidad. Es decir, el producto va evolucionando con cada una de las entregas previstas hasta que se amolda a lo requerido por el cliente o destinatario.

En el modelo incremental las tareas están divididas en iteraciones, es decir, pequeños lapsos en los cuales se trabaja para conseguir objetivos específicos. Las fases del modelo incremental son las siguientes:

- Requerimientos: son los objetivos centrales y específicos que persigue el proyecto.
- Definición de las tareas y las iteraciones: teniendo en cuenta lo que se busca, el siguiente paso es hacer una lista de tareas y agruparlas en las iteraciones que tendrá el proyecto. Cada una debe perseguir objetivos específicos que la definan como tal.
- Diseño de los incrementos: establecidas las iteraciones, es preciso definir cuál será la evolución del producto en cada una de ellas. Cada iteración debe superar a la que la ha precedido. Esto es lo que se denomina incremento.
- Desarrollo del incremento: posteriormente se realizan las tareas previstas y se desarrollan los incrementos establecidos en la etapa anterior.
- Validación de incrementos: al término de cada iteración, los responsables de la gestión del proyecto deben dar por buenos los incrementos que cada una de ellas ha arrojado. Si no son los esperados o si ha habido algún retroceso, es necesario volver la vista atrás y buscar las causas de ello.
- Integración de incrementos: una vez son validados, los incrementos dan forma a lo que se denomina línea incremental del proyecto en su conjunto. Cada incremento ha contribuido al resultado final.
- Entrega del producto: cuando el producto en su conjunto ha sido validado y se confirma su correspondencia con los objetivos iniciales, se procede a su entrega final.

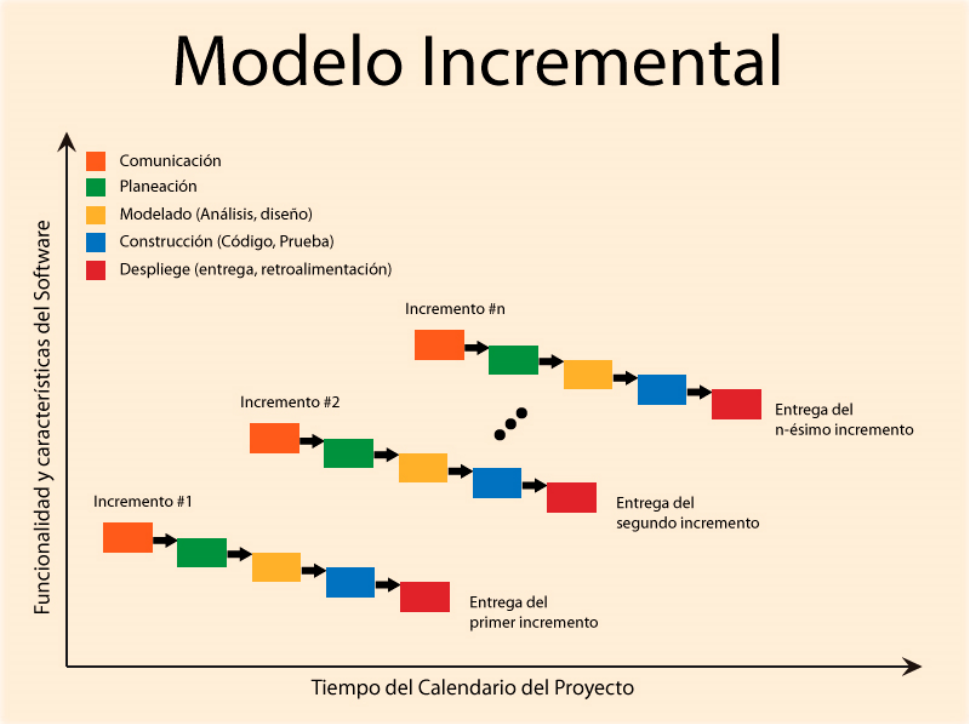


Figura 3. Modelo incremental.

- **Desarrollo rápido de aplicaciones (RAD).**

Destaca por su flexibilidad y adaptabilidad a los cambios. El producto es fácil de trasladar a otros entornos. Además, cada etapa del proyecto prioriza las necesidades de los clientes.

La metodología requiere de tiempos de entrega precisos para funcionar. No se aconseja utilizar en proyectos de menor escala. Así mismo, es indispensable la participación de expertos tanto de diseño como de programación.

1.7.2 Metodologías ágiles.

Las metodologías ágiles se caracterizan por su versatilidad debido a lo sencillo que les resulta el adaptarse a los cambios sin alterar las características originales del proyecto en el proceso.

En principio, estas surgen como respuesta a las carencias de las metodologías tradicionales, sin embargo, gracias a los constantes cambios en los mercados y los entornos de trabajo han ido evolucionando, en favor de un proceso de desarrollo optimizado. Las principales metodologías ágiles son las siguientes:

- **Programación extrema (XP).**

Las constantes pruebas al producto, garantizan un código de mayor calidad. El cliente puede participar activamente en el proceso. Así mismo, cualquier error

emergente es resuelto en el acto. La velocidad con la que se trabaja dentro de esta metodología complica la generación de documentación.

- **Kanban.**

Al implementar esta metodología se garantiza una constante supervisión del rendimiento tanto del equipo como del producto. Se mantienen los tiempos de producción estables, evitando los excesos. Por lo tanto, es fácil detectar elementos problemáticos en el proyecto.

- **Scrum.**

Constante retroalimentación por parte del equipo de trabajo y los clientes. Posee calendarios de entrega y supervisión bien definidos. Además, las actividades se clasifican de acuerdo a su importancia.

En esta metodología es precisa una planificación exhaustiva de las tareas y fechas de entrega del proyecto. Los expertos participantes deben estar sobrecalificados en sus respectivos ámbitos. No es aconsejable utilizarlos en proyectos a gran escala.

- **Lean.**

Con ayuda de esta metodología se optimizan los costos de producción y la velocidad de entrega del proyecto. Cualquier elemento sin valor para el cliente es eliminado. Así mismo, sobresale por su equipo motivado y ambiente de trabajo colaborativo.

Se requiere de expertos en todos los ámbitos que el proyecto dicte necesitar. n Los costos de desarrollo del proyecto son muy elevados.

- **Desarrollo basado en características (FDD).**

En esta metodología, cualquier elemento sin valor para el cliente es eliminado. Cada elemento incorporado dentro del proyecto cumple con sus requerimientos. Además, se garantiza un eficiente control del tiempo y la calidad del producto.

Similar a otras metodologías, se tiene una fuerte dependencia al equipo de trabajo del proyecto. Se precisa de un experto en programación que cumpla el papel de líder del proyecto. Por su parte, la documentación del proyecto es casi inexistente.

1.7.3 Elección de la metodología.

Una vez se ha revisado todas las metodologías, se ha optado por el modelo incremental para la realización de este TFM. Se ha elegido este modelo debido a que permite implementar diferentes iteraciones, con objetivos bien definidos, a lo largo del

proyecto, incorporando en etapas intermedias la corrección de errores. De este modo, se mantiene un control sobre el desarrollo del proyecto.

2. Diseño.

En este apartado se presenta el diseño del prototipo del sistema de monitorización de la calidad del aire en interiores de bajo coste que se propone en este TFM.

Para ello, en primer lugar, se describirá la arquitectura IoT utilizada para el desarrollo de este sistema. En segundo lugar, se explicará los dispositivos, sensores y placas de desarrollo que intervienen en el proyecto. A continuación, se explicará las claves del diseño de los soportes utilizados para el posicionamiento tanto de la batería externa como la protoboard, que contiene el MPU6050. Por último, se comentará todo lo relacionado con el desarrollo del software.

2.1 Arquitectura IoT del sistema de monitorización de la calidad del aire en interiores.

En este apartado se presenta la arquitectura IoT y componentes utilizados en el sistema de monitorización de la calidad de aire en interiores implementada.

El sistema IoT implementado en este TFM contiene los siguientes elementos:

- **Sensor de gas SEN0159**, es el sensor encargado de medir las concentraciones de CO_2 en el ambiente.
- **Unidad de medición inercial MPU6050**, encargada de medir los ángulos de apertura tanto de la puerta como de la ventana. Por lo tanto, se utilizará dos unidades, una para la ventana y otra para la puerta.
- **NodeMCU Amica (ESP8266)**, placa de desarrollo encargada de recibir los datos de la unidad de medición inercial MPU6050 y transmitirlos mediante MQTT al nodo sumidero (Raspberry Pi). También se utilizará dos unidades, una para la ventana y otra para la puerta.
- **ESP-WROOM 32D (ESP32)**, placa de desarrollo encargada de recibir los datos del sensor de gas y transmitirlos mediante MQTT al nodo sumidero (Raspberry Pi).
- **Raspberry Pi 3 Model B**, es el nodo sumidero que recibirá los datos de los diferentes sensores y los almacenará en una base de datos para

tener persistencia en los datos. Además, se encuentra alojada la plataforma IoT.

- **Soportes**, diseñados con el objetivo de posicionar tanto la batería externa como la protoboard, que contiene el MPU6050, en la puerta y en la ventana.
- **Powerbank**, encargada de suministrar energía a las placas de desarrollo NodeMCU Amica, utilizadas tanto en la ventana como en la puerta. Por lo tanto, también se utilizará dos unidades, una para la ventana y otra para la puerta.
- **Base de datos (MongoDB)**, es la base datos que almacena los datos provenientes de los diferentes sensores, para su posterior análisis.
- **Plataforma IoT (Home Assistant)**, permite la visualización gráfica, en tiempo real, de los datos provenientes de los diferentes sensores.

En la siguiente figura se muestra un esquema de la arquitectura IoT que integra el prototipo diseñado para la monitorización de la calidad de aire en interiores:

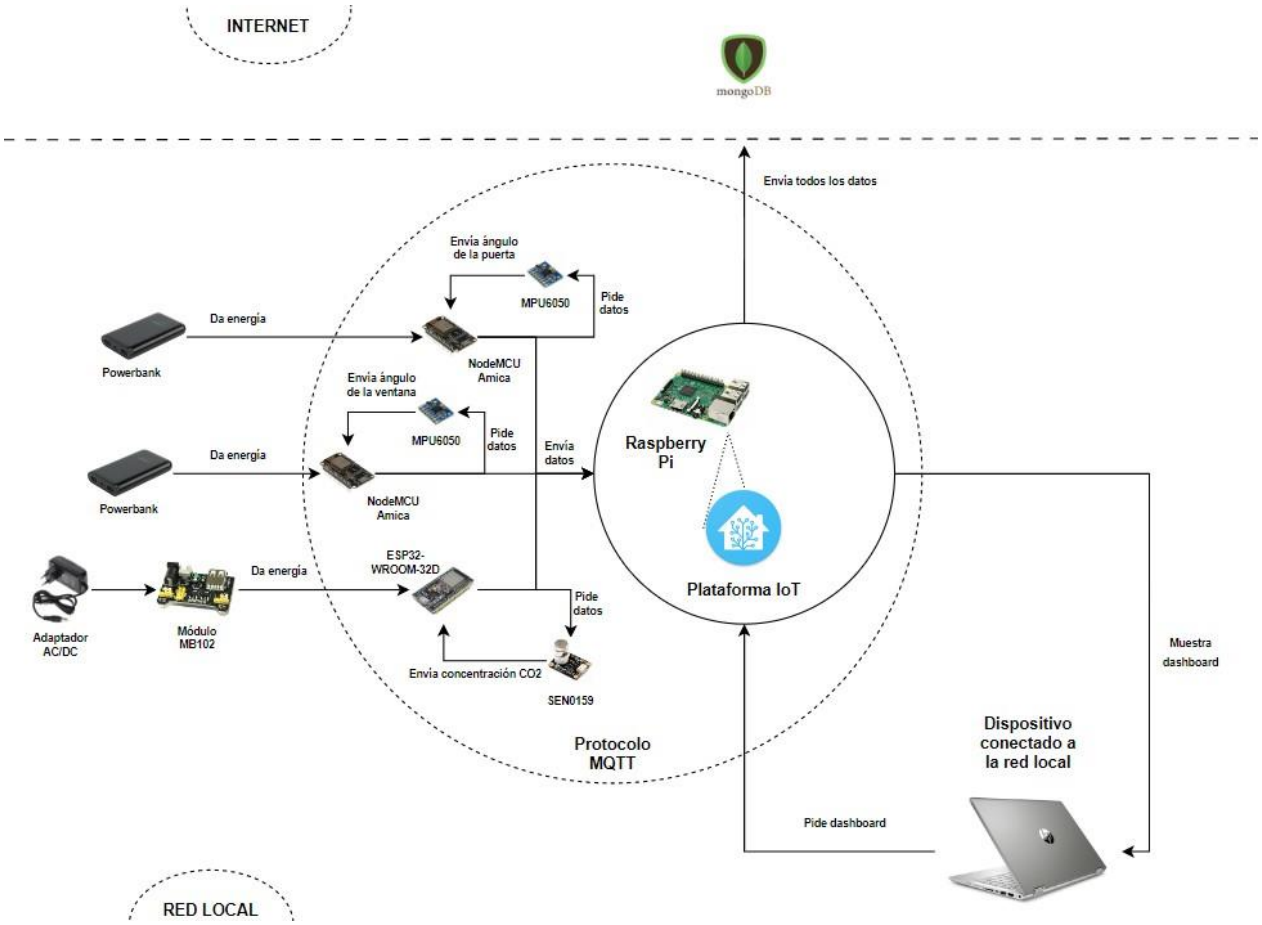


Figura 4. Arquitectura IoT.

2.2 Diseño del modelo 3D.

Para la calibración de los sensores MPU6050, utilizados tanto en la ventana como en la puerta, estos deben permanecer colocados en la misma posición con el objetivo de obtener los offsets de la velocidad de giro, que a posteriori, se utilizará en el software diseñado para la autocalibración de estos sensores. Para obtener correctamente estos offsets, los sensores MPU6050 deben estar colocados de tal forma que el eje Z coincida con la gravedad. Ante esta situación, se ha decidido diseñar un soporte que permita posicionar la protoboard, que contiene al sensor MPU6050, tanto en la puerta como en la ventana como se muestra en la siguiente figura:

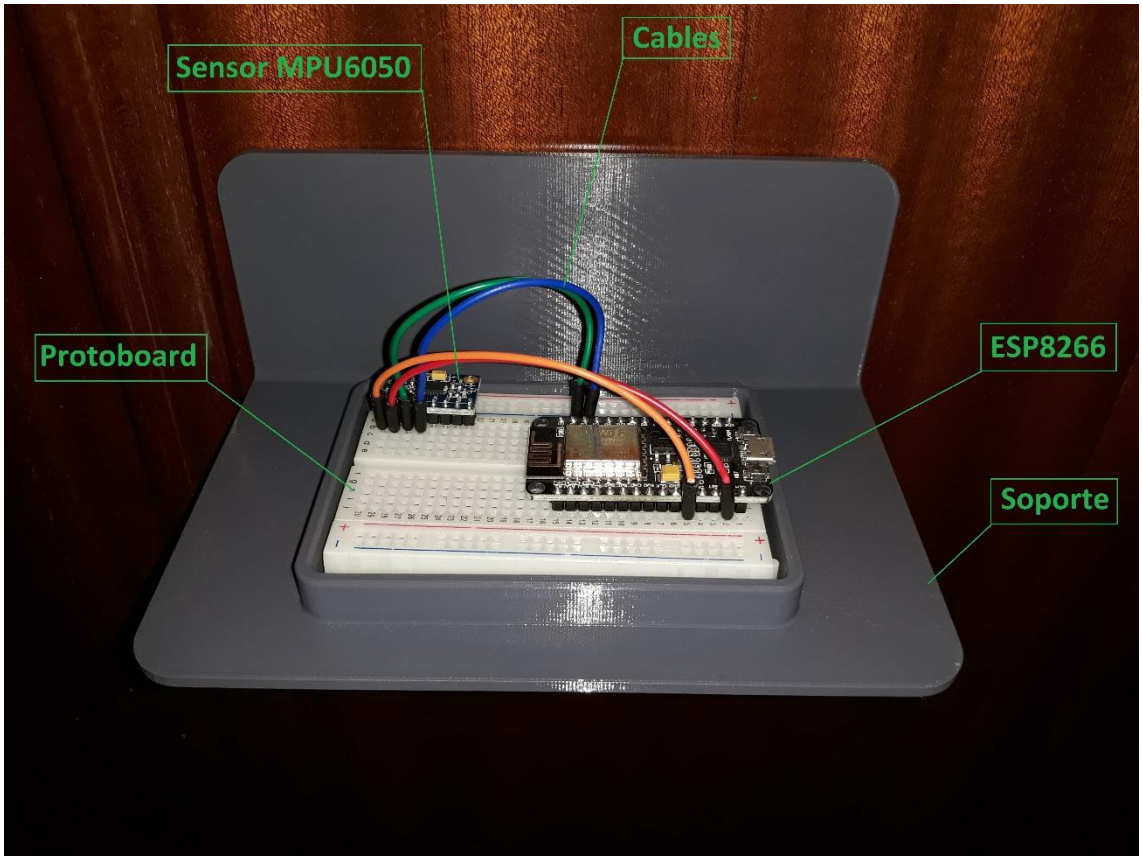


Figura 5. Soporte diseñado para el posicionamiento del sensor MPU6050.

Además, se ha diseñado un segundo soporte para colocar la batería externa que proporcionará energía a la placa de desarrollo ESP8266, como se muestra en la siguiente figura:

Figura 6. Soporte diseñado para la colocación de la powerbank.

Para el diseño de ambos soportes se ha realizado un prototipo en impresión 3D. La impresión 3D es el proceso de creación de objetos tridimensionales mediante el depósito de capas de material unas sobre otras. Está técnica ha sido una revolución en el sector de la ingeniería debido a que permite crear prototipos muy complejos a una gran velocidad. Aunque la impresión de un objeto puede llevar horas o incluso días enteros, sigue siendo mucha más rápida que los métodos de producción habituales, como el moldeo por inyección.

El diseño de ambos soportes se realizó en CATIA v5. En la universidad de Jaén disponemos de licencia gratuita por el hecho de ser estudiante. Esta aplicación permite diseñar modelos y exportarlos a formato STL (Standar Tessellation Language) que es el utilizado para la herramienta de impresión 3D. Es un software de diseño 3D muy potente utilizado mucho en la industria, sobre todo en automoción. En las siguientes figuras se muestras los modelos 3D diseñados en CATIA v5:

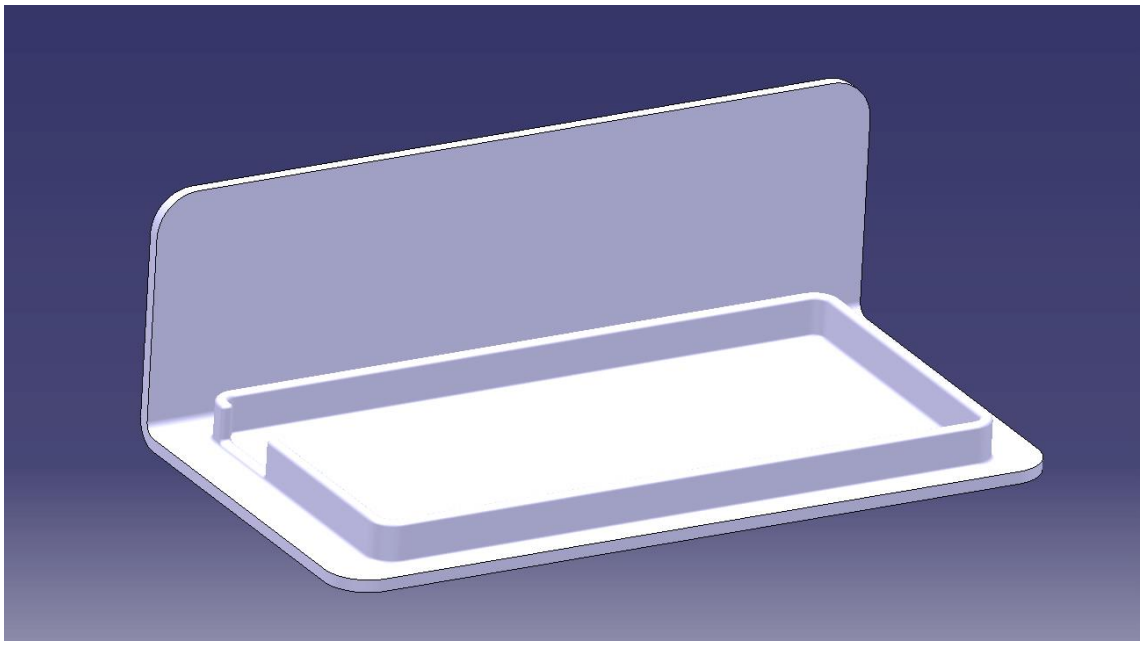


Figura 7. Modelo 3D del soporte diseñado para la colocación de la powerbank.

Figura 8. Modelo 3D diseñado para el posicionamiento del sensor MPU6050.

Para el diseño del soporte de la figura 6 se ha tenido en cuenta las dimensiones de la powerbank con una tolerancia de 1 mm, encajando así, sin problemas, la powerbank en el hueco destinado a ella. Además, en una de las paredes, donde se encaja la powerbank, hay una abertura con el objetivo de permitir la introducción del cable USB en la batería externa.

Para el diseño de soporte de la figura 7 se ha tenido en cuenta las dimensiones de la protoboard con una tolerancia de 0.5 mm. Aquí la tolerancia es menor ya que se ha intentado que la protoboard encaje a la perfección en el hueco destinado a ella, sin posibilidad a ningún movimiento. Esto permitirá una mejor calibración del sensor MPU6050, como ya se ha explicado anteriormente.

2.3 Diseño hardware.

En este apartado se muestra todos los componentes que se han tenido en cuenta para el diseño de la arquitectura IoT del prototipo de monitorización de la calidad del aire en interiores.

2.3.1 Placas de desarrollo.

Las placas de desarrollo son dispositivos que cuentan con un microcontrolador reprogramable, el cual, puede ejecutar instrucciones para un fin específico. Generalmente estas placas cuentan con entradas y salidas analógicas y/o digitales para permitir la comunicación con sensores externos.

Las placas de desarrollo utilizadas en este proyecto son las siguientes:

- **Placa de desarrollo NodeMCU Amica.**

Placa de desarrollo basada en ESP8266. Se trata de la segunda versión de la placa NodeMCU. Respecto a la primera versión, llamada devkit, esta placa monta un ESP12E en vez de un ESP12, por lo que tiene más pines disponibles que el modelo original. Además, es más estrecha que la versión anterior, tapando solo 8 hileras de una protoboard. Esto deja una hilera adicional a cada lado para realizar conexiones.

Figura 9. Placa de desarrollo NodeMCU Amica.

Otras ventajas de esta placa de desarrollo que podemos destacar son las siguientes:

- Tiene puerto micro USB y conversor Serie-USB.
- Programación sencilla a través del micro USB.
- Alimentación a través de USB.
- Terminales (pines) para facilitar la conexión.
- LED y botón de reset integrado.
- Wi-Fi: 802.11 b/g/n.

Lo más adecuado para trabajar con una placa de desarrollo es conectarla a una protoboard. Por ello, una de las principales razones de la elección de esta placa de desarrollo, sobre las demás versiones, para su conexión con el sensor MPU6050 es debido a que se trata de la versión más estrecha. Deja una hilera adicional a cada lado para realizar conexiones. Esto nos permitirá realizar las conexiones de una manera más cómodo y nos dará estabilidad.

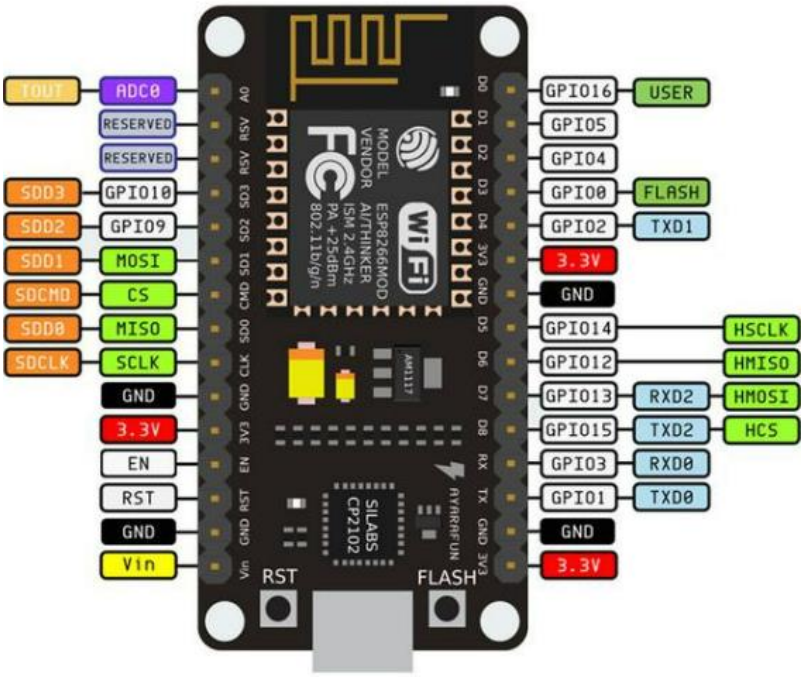


Figura 10. Pines de la placa de desarrollo NodeMCU Amica.

Otro motivo para la elección de esta placa de desarrollo es que tiene integrado un módulo Wifi, que nos permite transmitir los datos sin necesidad de adquirir un hardware adicional.

- **Placa de desarrollo ESP32-WROOM-32D.**

La ESP32 es una placa de desarrollo de bajo costo y alto rendimiento que combina un microcontrolador de doble núcleo a 240 MHz con Wi-Fi y Bluetooth integrados.

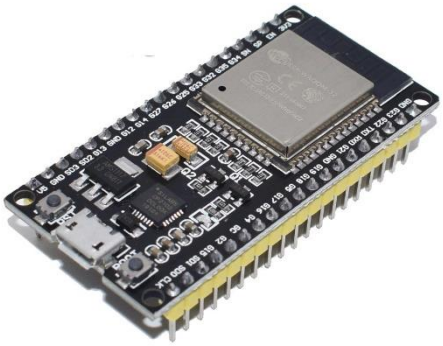


Figura 11. Placa de desarrollo ESP32-WROOM-32D.

Otras características de esta placa de desarrollo son las siguientes:

- Wi-Fi: 2.4GHz 802.11 g/b/n.
- Soporte para protocolos de red como TCP/IP, UDP, HTTP, HTTPS, MQTT y WebSocket.
- 34 puertos GPIO para una mayor flexibilidad en la conexión de componentes externos.
- 18 canales ADC de 12 bits.
- Memoria flash de hasta 4MB.

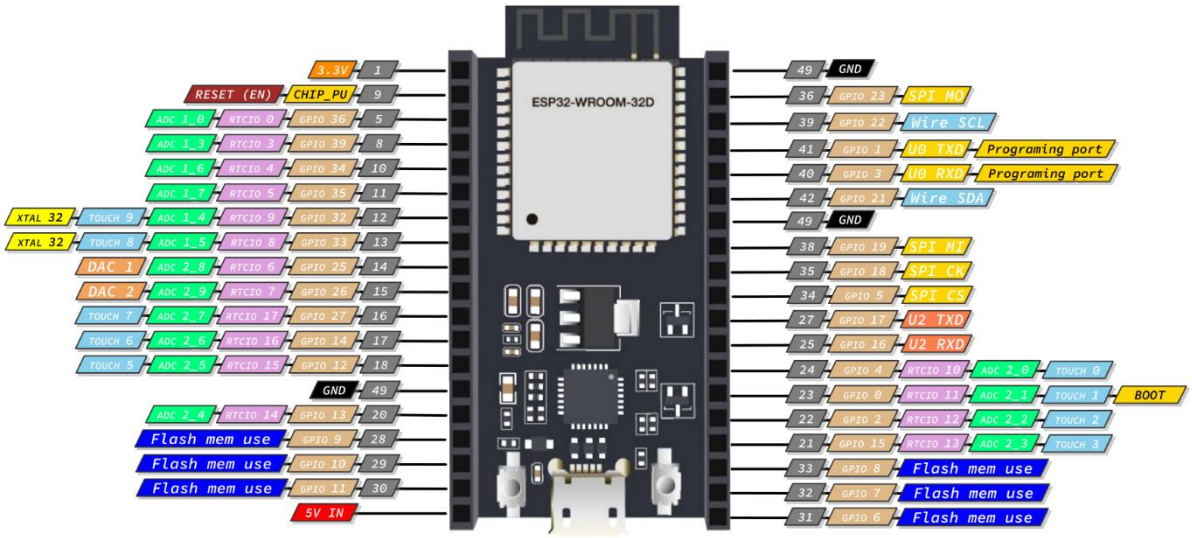


Figura 12. Pines de la placa de desarrollo ESP32-WROOM-32D.

El principal motivo para la elección de esta placa de desarrollo para su conexión con el sensor de gas es la inclusión de 18 canales ADC de 12 bits. En otras placas, como en la ESP8266, el Wi-Fi utiliza el único pin ADC disponible para medir voltajes internos y así ajustar su potencia de salida y eso hace que los resultados leídos por el sensor de gas no sean correctos. En cambio, esta placa dispone pines ADC donde el Wi-Fi no interviene por lo que los resultados obtenidos por el sensor de gas son más estables.

- **Raspberry Pi 3 Model B**

Se trata de un microcontrolador con mayor capacidad de cómputo que una ESP8266. Además, permite la instalación de un sistema operativo, que, en nuestro caso, se ha utilizado la versión Lite de 64 bytes que no dispone de entorno gráfico y la hace más rápida.

Figura 13. Raspberry Pi 3 Model B.

Algunas ventajas que tiene esta placa de desarrollo y que han sido clave en su elección son las siguientes:

- Procesador: Broadcom BCM2837 de 1200 MHz ARM quad core Cortex-A53 de cuatro núcleos y con juego de instrucciones ARMv8 lo que implica capacidades parciales de 64 bits.
- GPU: VideoCore IV de doble núcleo a 400 MHz con soporte de Open GL ES 2.0, hardware acelerado Open VG hasta 1080p30 H.264.
- RAM: 1GB SDRAM LPDDR2.
- Almacenamiento: tarjeta microSD para el Sistema Operativo.
- Wi-Fi: 802.11 Wireless LAN.

2.3.2 Sensores.

Los sensores utilizados en este proyecto son los siguientes:

- **MPU6050**: El MPU-6050 es una unidad de medición inercial (IMU) de seis grados de libertad (6DOF) fabricado por Invensense, que combina un acelerómetro de 3 ejes y un giroscopio de 3 ejes. La comunicación puede realizarse tanto por SPI como por bus I2C, por lo que es sencillo obtener los datos medidos.

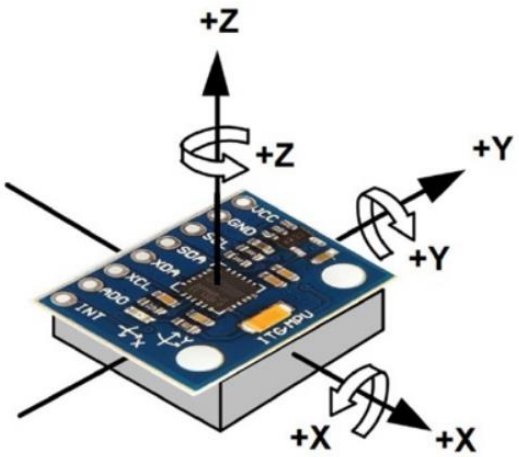


Figura 14. Sensor MPU6050.

El acelerómetro de este sensor permite medir la aceleración a través de la segunda ley de Newton, $a = \frac{F}{m}$, es decir, utilizan parámetros de fuerza y masa. Los acelerómetros tienen internamente un MEMS (MicroElectroMechanical Systems) que de forma similar a un sistema masa resorte permite medir la aceleración. El giroscopio permite medir la velocidad angular de un objeto, es decir, lo rápido que gira un objeto a través de su eje. En este caso, también se utilizan técnicas MEMS para medir dicha velocidad usando un efecto conocido como Coriolis.

En nuestro caso, se utilizará la velocidad en el eje Z obtenida en el giroscopio para obtener el ángulo de apertura de la puerta y de la ventana. Para ello utilizaremos las siguientes expresiones:

$$\theta_z = \theta_z + \omega_{giroscopio} * dt \quad (1)$$

$$dt = t - t_0 \quad (2)$$

donde θ_z es el ángulo de apertura, $\omega_{giroscopio}$ es la velocidad angular obtenida por el sensor MPU6050 y dt es el tiempo que transcurre en la apertura de la puerta de una posición a otra.

Se utiliza la velocidad en el eje Z obtenida en el giroscopio para el cálculo del ángulo de apertura debido a que el sensor MPU6050 debe estar siempre en una posición en la cual el eje Z coincida con la gravedad par su correcta calibración. Para la calibración de este sensor se ha utilizado el sketch MPU6050_Calibration, creado por Luis Ródenas. Este sketch nos proporciona unos offsets que se utilizan para la autocalibración del sensor. Por lo tanto, si el sensor debe estar en una posición horizontal sólo podemos utilizar la velocidad giro en el eje Z para el cálculo del ángulo de apertura.

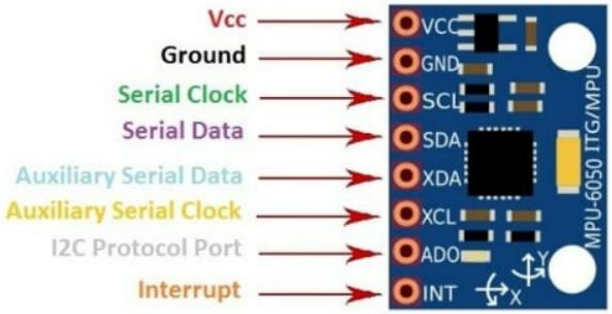


Figura 15. Pines del sensor MPU6050.

El diagrama de pines de este sensor se puede ver en la siguiente figura:Vcc: es el pin de voltaje de la fuente de alimentación.

- Ground: es la tierra de la fuente de alimentación.
- Serial Clock: es el reloj de serie I2C.
- Serial Data: es el pin de datos en serie de I2C.
- Auxilary Serial Data: Es el pin de datos de la serie maestra de I2C. Este pin se utiliza para conectar sensores externos.
- Auxilary Serial Clock: es el reloj serial maestro de I2C. Esta clavija se usa para conectar sensores externos.
- I2C Protocol Port: Es el pin LSB de dirección esclavo de I2C.

- **SEN0159**: Es un sensor, desarrollado por DFRobot, que permite medir el nivel del CO_2 en el ambiente. El voltaje de salida del módulo disminuye conforme aumenta la concentración de CO_2. Tiene un potenciómetro que permite establecer el voltaje umbral. Mientras la concentración de CO_2 sea lo

suficientemente alta (la tensión es inferior al umbral), se emitirá una señal digital (ON/OFF). Además, tiene un sensor de gas MG-811 a bordo que es altamente sensible al CO_2 y menos sensible al alcohol y CO.

Figura 16. Sensor de gas SEN0159.

Según la gráfica de sensibilidad extraída de la hoja de características del sensor, figura 16, se puede observar que el rango de medición del nivel de CO2 de este sensor es de 400 ppm (partes por millón) a 10000 ppm.

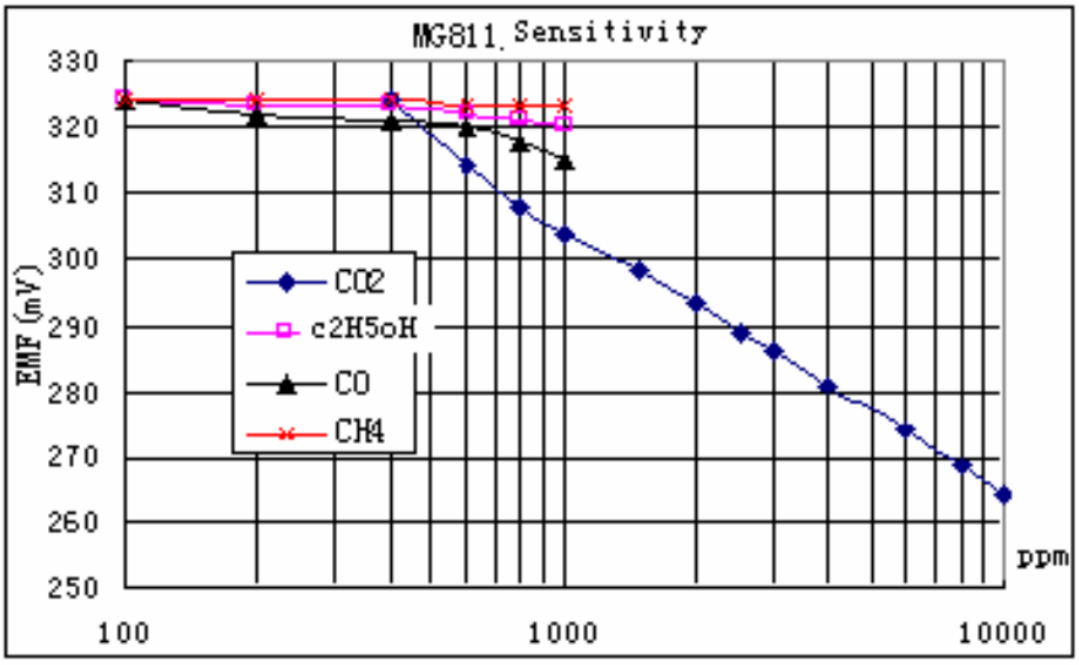


Figura 17. Gráfica de sensibilidad del sensor de gas.

Una consideración a tener en cuenta es el tiempo de precalentamiento. Este módulo es un sensor electroquímico por lo que se necesitará calibrarlo antes de realizar cualquier medición. Para ello, debemos poner el módulo en un área con aire limpio. Después de trabajar durante 48 horas [23], medimos el voltaje de salida del módulo. Ese valor se utilizará para calcular el ZERO_POINT_VOLTAGE que se utiliza

en el código del programa utilizado. ZERO_POINT_VOLTAGE se calcula dividiendo el voltaje medido entre 8.5.

Para recoger los datos de manera eficiente se debe tener en cuenta que, en la placa ESP32-WROOM-32D, el pin ADC sólo permite un rango de entrada de 0-3.3V. Sin embargo, la señal analógica que emite el sensor SEN0159 es de un rango 0-5V, por lo tanto, hay que adaptar la señal antes de introducirla en el pin ADC de la placa ESP32-WROOM-32D. Para solucionar este problema, se ha diseñado un divisor de tensión que divide la tensión de entrada de 5V a 2.5V.

A continuación, se explica el proceso llevado a cabo para el diseño de este divisor de tensión ayudándonos en la siguiente figura:

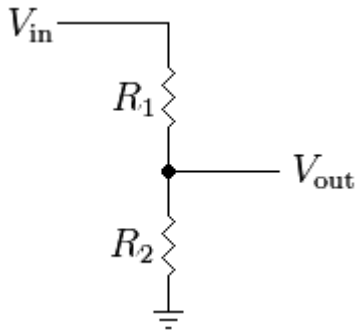


Figura 18. Esquema de un divisor de tensión.

En la figura 17, se muestra el esquema de un divisor de tensión, donde:

- V_{in} es la tensión de entrada y se mide en voltios (V). En nuestro caso particular, esta tensión de entrada tiene un valor de 5V, que es la tensión máxima de la señal analógica que emite el sensor SEN0159.
- R_1 y R_2 son las resistencias que debemos modelar para el diseño del divisor de tensión y se miden en ohmios (Ω).
- V_{out} es la tensión de salida y se mide en voltios (V). En nuestro caso, se ha decido que esta tensión de salida tenga un valor de 2.5V. Este valor se encuentra en el rango de entrada que permite el pin ADC de la placa ESP32-WROOM-32D.

Sabiendo estos datos, debemos calcular el valor de las resistencias R_1 y R_2. Para ello, partimos de la ley de Ohm:

$$I = \frac{V_{in}}{R_t} \quad (3)$$

donde:

- I es la intensidad que circula por el divisor de tensión y se mide en Amperios (A).
- V_{in} es la tensión de entrada, como se ha comentado anteriormente.
- R_t es la resistencia total del circuito, es decir, la suma de R_1 y R_2.

Además, también con la ley de ohm podemos sacar que el valor de la tensión de salida, V_{out}, es:

$$V_{out} = I * R_2 \quad (4)$$

Utilizando las ecuaciones (1) y (2) podemos sacar la siguiente expresión matemática:

$$\left.\begin{array}{l} I = \frac{V_{in}}{R_t} = \frac{V_{in}}{R_1 + R_2} \\ I = \frac{V_{out}}{R_2} \end{array}\right\} \quad \frac{V_{in}}{R_1+R_2} = \frac{V_{out}}{R_2} \quad (5)$$

Se fija la resistencia R_1 a un valor adecuado como puede ser 10kΩ, que no tenga ni muy baja ni muy alta impedancia. Ahora, se calcula el valor de la resistencia R_2 a partir de la ecuación (5):

$$R_2 = \frac{R_1}{\frac{V_{in}}{V_{out}} - 1} = \frac{10000}{\frac{5}{2.5} - 1} = 10000\Omega = 10\text{k}\Omega$$

2.3.3 Otros.

Además de las placas y sensores ya explicados, en este proyecto también se ha utilizado los siguientes componentes:

- **Fuente de alimentación MB102:** Es un módulo de alimentación que se utiliza comúnmente para proyectos de electrónica y prototipado. En nuestro proyecto, se ha utilizado para alimentar la placa ESP32-WROOM-32D con 5V. A continuación, se presenta algunas de sus características:
  - Voltaje de entrada: Esta fuente de alimentación tiene un voltaje de entrada que va desde 6V a 12V DC.
  - Voltaje de salida: Esta fuente de alimentación dos salidas de voltaje regulables 3.3V y 5V DC respectivamente. También tiene una salida de voltaje fija de 1.5 V, que se utiliza comúnmente para alimentar sensores y otros componentes.

- Corriente máxima: La corriente máxima que la fuente de alimentación MB102 puede suministrar es de 700 mA para la salida de 3.3V, y de 1A para la salida de 5V.
- Protección contra cortocircuitos.

Figura 19. Fuente de alimentación MB102.

Esta fuente de alimentación se ha utilizado para alimentar de forma estable al sensor de gas SEN0159. Este módulo, combinado con el adaptador AC/DC, nos permite alimentar al sensor de gas a partir de la corriente eléctrica doméstica por lo que conseguimos una alimentación constante. El sensor de gas SENS0159 es un sensor electroquímico, como se ha comentado anteriormente, por lo que necesita, según el fabricante, una fase de precalentamiento de 48 horas para su correcto funcionamiento. Esto quiere decir que el sensor, como mínimo, va a estar funcionando durante 2 días sin interrupción por lo que su alimentación debe ser estable. De ahí, la elección de este módulo frente a, por ejemplo, a una powerbank. Además, se trata de un sensor muy sensible por lo que una inestabilidad en la alimentación conllevaría a que nos proporcionase valores erróneos del nivel de CO_2.

- **Adaptador AC/DC**: Es un tipo de alimentación externa, que, en nuestro proyecto, permitirá alimentar la fuente de alimentación MB102 con 9V DC a partir de la tensión doméstica de 220V AC.

Figura 20. Adaptador AC/DC.

2.3.4 Diseño del circuito electrónico.

En este apartado se mostrará el esquemático que se ha utilizado para el montaje de todos los sensores y placas de desarrollo. En la siguiente figura se puede observar el esquemático del montaje utilizado para el funcionamiento de los sensores MPU6050:

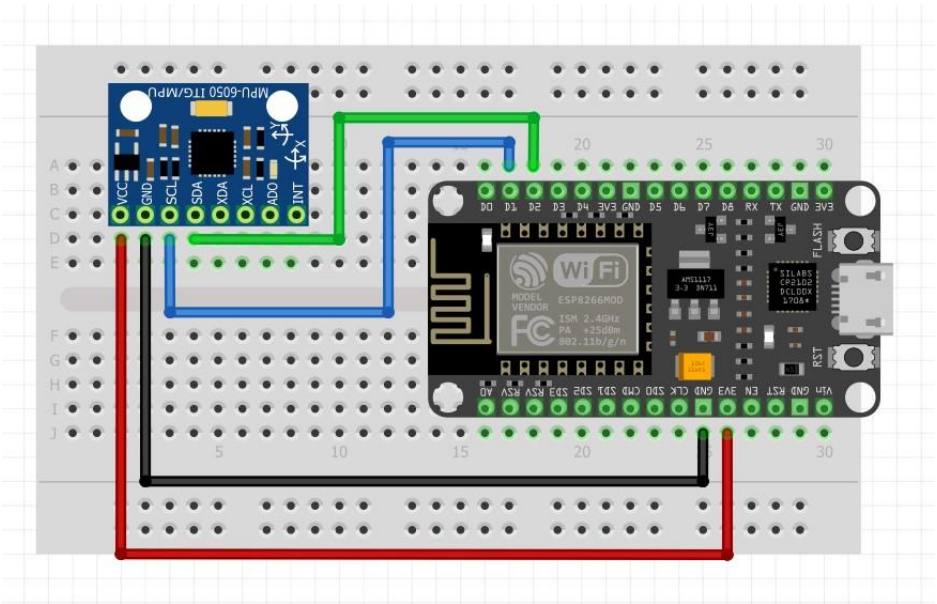


Figura 21. Esquemático del montaje del sensor MPU6050.

El sensor MPU6050 se alimenta a través de VCC mediante un voltaje de 3.3 V. Además, la tierra del MPU6050 debe ir a la tierra de la placa ESP8266. La placa de desarrollo y el sensor se comunican mediante el bus I2C. El I2C precisa de dos líneas de señal: reloj (SCL, Serial Clock) y la línea de datos (SDA, Serial Data). Por ello, se

debe conectar el SCL del sensor con el SCL de la placa de desarrollo y el SDA del sensor con el SDA de la placa de desarrollo. En la placa de desarrollo NodeMCU Amica el Serial Clock (SCL) se encuentra en el pin D1 y el Serial Data (SDA) se encuentra en el pin D2.

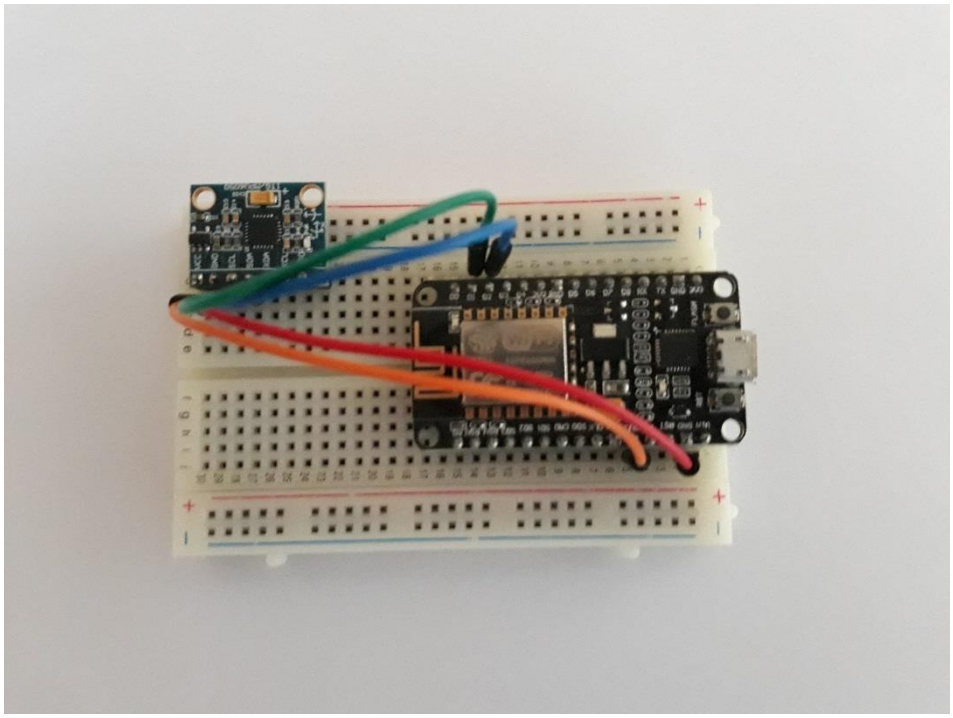


Figura 22. Montaje del sensor MPU6050 en la protoboard.

Para la conexión del sensor de gas se ha utilizado el pin analógico A0 por ser el responsable de proporcionar las medidas RAW y no se ha utilizado un pin digital. A continuación, se muestra el esquemático utilizado en el montaje de este sensor:

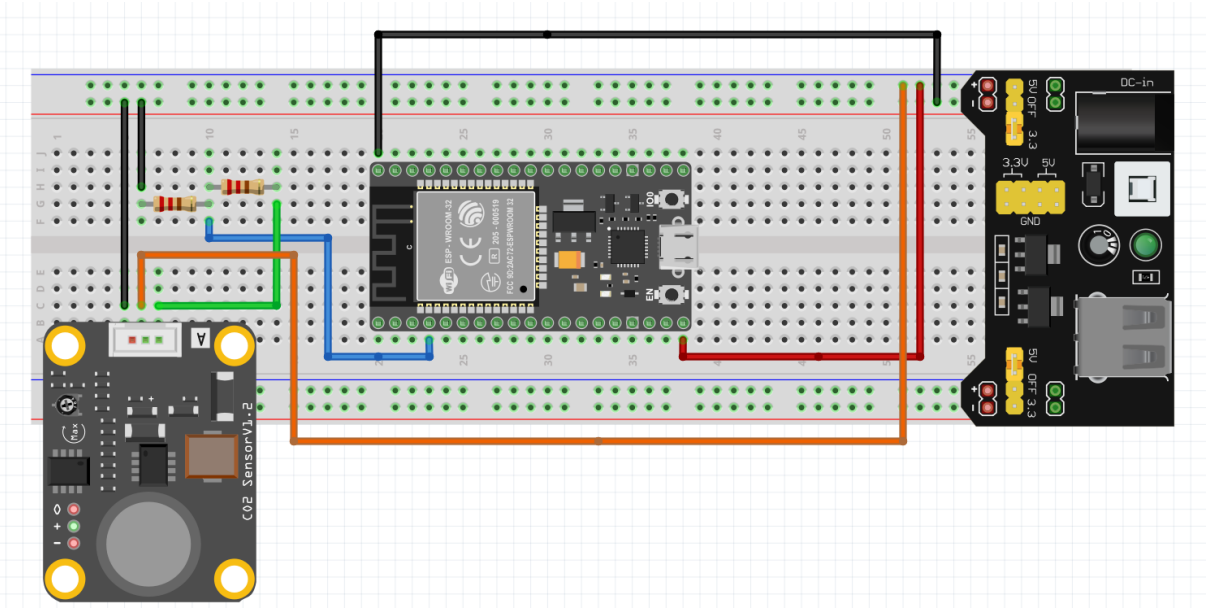


Figura 23. Esquemático del montaje del sensor SEN0159.

En la figura 23 se puede observar cómo se ha conectado el sensor de gas con el pin ADC de la placa de desarrollo a través de un divisor de tensión, imprescindible como se ha comentado en apartados anteriores. La placa MB102 será la encargada de proporcionar 5V de forma continua tanto a la placa de desarrollo como al sensor de gas. Además, se observa que las tierras del sensor de gas, la placa de desarrollo y la fuente de alimentación están conectadas entre sí.

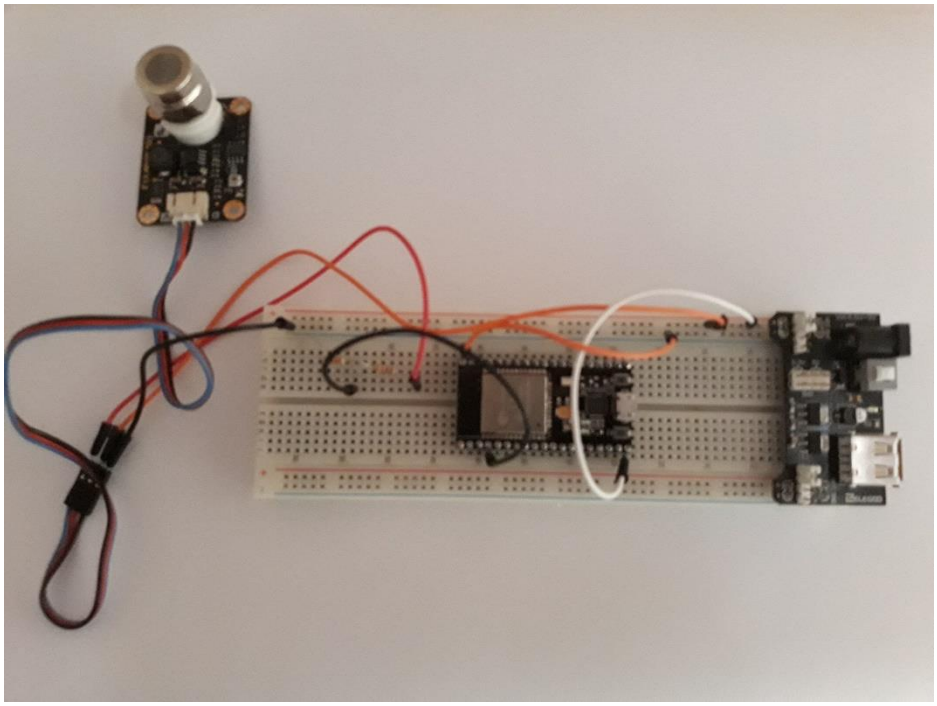


Figura 24. Montaje del sensor SEN0159 en la protoboard.

2.4 Diseño del software.

En esta sección se explica el proceso que se ha llevado a cabo para cumplir las especificaciones software que han llevado a la creación del sistema IoT monitorización de la calidad del aire en interiores a cumplir con los objetivos.

2.4.1 Diagramas de flujo.

Los diagramas de flujo son una representación gráfica y secuencial de un proceso o flujo de trabajo con todas las tareas y actividades principales necesarias para lograr un objetivo común (Asana, 2022). En nuestro caso, se utilizarán para explicar el comportamiento tanto del sensor MPU6050, instalado en la ventana y en la puerta de la habitación, como del sensor de gas.

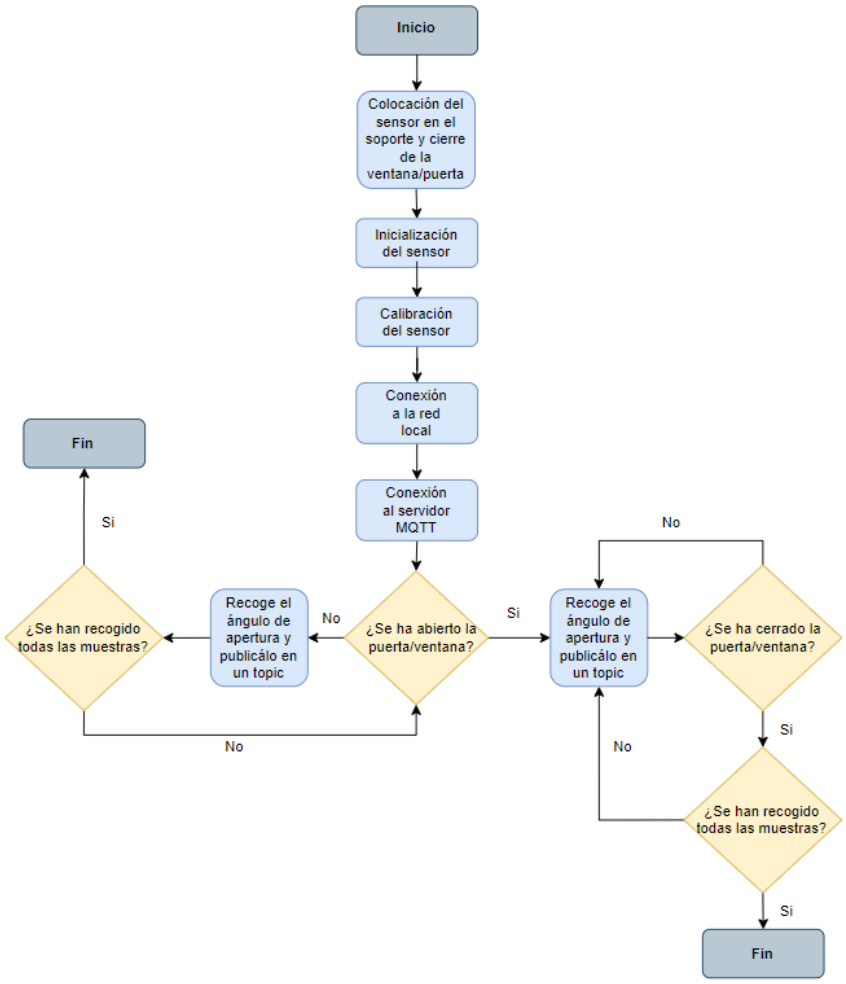


Figura 25. Diagrama de flujo del MPU6050.

Es muy importante que el sensor se mantenga en la misma posición cuando se hace la calibración de este. Por ello, se ha diseñado un soporte en el que colocar la protoboard que aloja al sensor MPU6050. Además, en nuestro caso, esta calibración se realiza con la ventana cerrada. Tras la calibración, la placa de desarrollo se conecta con la red local y el servidor MQTT. Si la ventana/puerta no se abre, se recogerá los datos del ángulo de apertura y se publicarán en un topic hasta que se tenga todas las muestras necesarias. Si la ventana/puerta se abre, se recogerá los datos del ángulo de apertura y se publicarán en un topic hasta que se cierre la ventana y se hayan recogido todas las muestras necesarias.

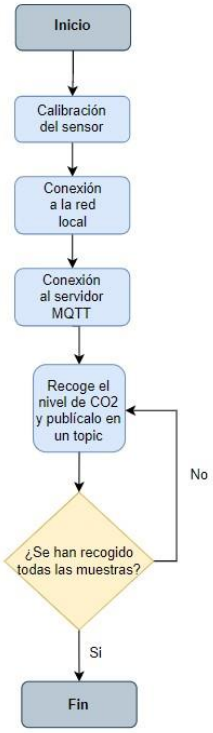


Figura 26. Diagrama de flujo del SEN0159.

Una vez se ha calibrado el sensor de gas como se ha explicado en apartados anteriores, la placa de desarrollo, que está conectada al sensor, se conecta a la red local y al servidor MQTT. A continuación, el sensor recoge los datos del nivel de CO_2 que hay en la habitación y los publica en un topic hasta que se han recogido todas las muestras necesarias.

2.4.2 Casos de uso.

En este apartado tendremos en cuenta las acciones que el analista y administrador puede realizar sobre el sistema. Para ello, nos apoyaremos en los diagramas de casos de uso. Un diagrama de uso es una representación visual de las diferentes maneras y posibles escenarios de usar un sistema.

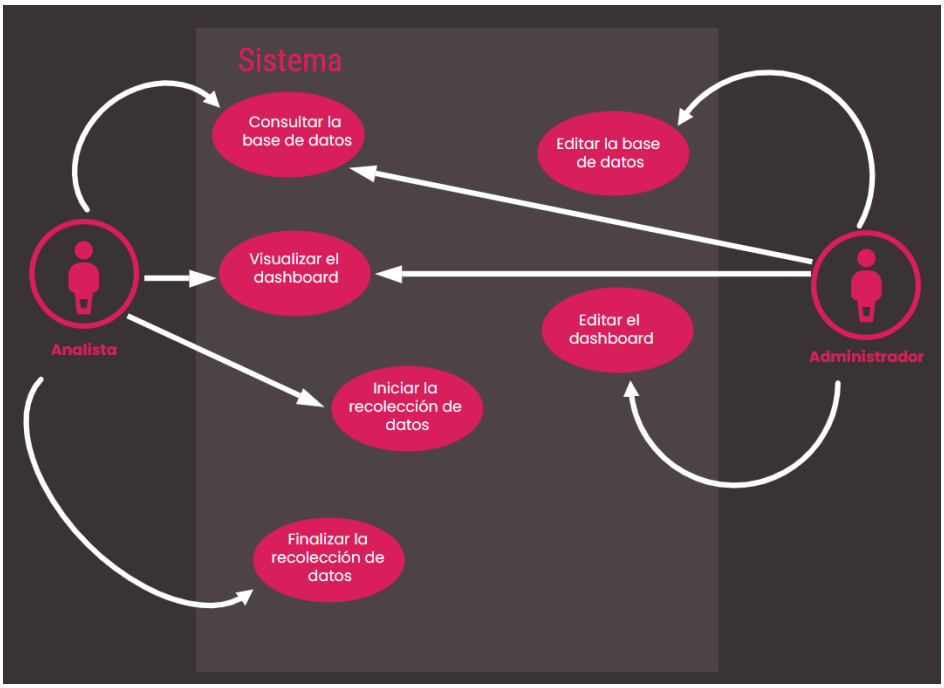


Figura 27. Diagrama de casos de uso.

En las siguientes tablas se muestras los casos de uso que se han definido para este sistema.

CDU-1	Consultar la base de datos
Participantes	Analista/Administrador
Descripción	Analista y administrador pueden visualizar la base de datos.
Condición anterior	Existir una base de datos en MongoDB.
Flujo natural	1- Iniciar sesión en MongoDB Compass. 2-Acceder a la base de datos.
Flujo alternativo	1- Conectarse mediante la línea de comandos. 2-Realizar consultas a la base de datos.
Condición posterior	Ninguna
Observaciones	Ninguna

Tabla 8. CDU-1 Consultar la base de datos.

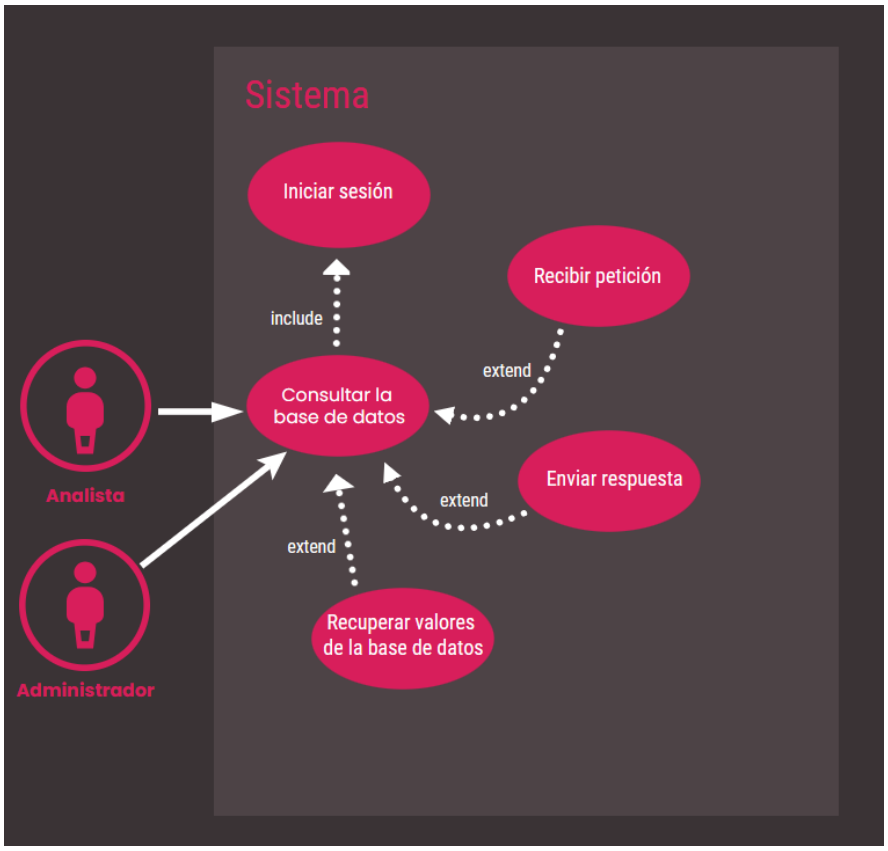


Figura 28. Diagrama caso de uso 1.

CDU-2	Visualizar el dashboard
Participantes	Analista/Administrador
Descripción	Analista y administrador pueden visualizar el dashboard.
Condición anterior	Existir un dashboard creado en la plataforma IoT (Home Assistant).
Flujo natural	1- Iniciar sesión en la plataforma IoT. 2-Acceder al dashboard.
Flujo alternativo	No existe flujo alternativo
Condición posterior	Ninguna
Observaciones	Ninguna

Tabla 9. CDU-2 Visualizar el dashboard.

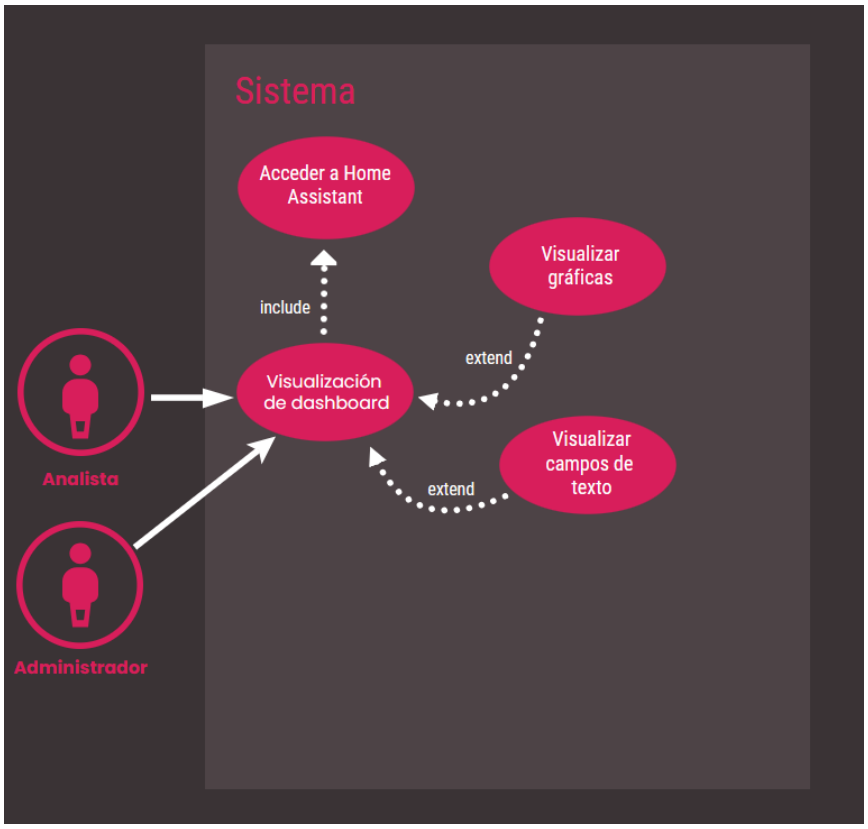


Figura 29. Diagrama caso de uso 2.

CDU-3	Iniciar la recolección de datos
Participantes	Analista
Descripción	El analista puede iniciar la recolección de datos en la base de datos.
Condición anterior	Se ha creado el proyecto en Python para la recolección de datos, está alojado en la Raspberry Pi y está configurado para que se inicie al arrancar la Raspberry Pi.
Flujo natural	1-Conectar la Raspberry Pi a la corriente eléctrica. El proyecto creado en Python se ejecutará automáticamente.
Flujo alternativo	No existe flujo alternativo
Condición posterior	Ninguna
Observaciones	Ninguna

Tabla 10. CDU-3 Iniciar la recolección de datos.

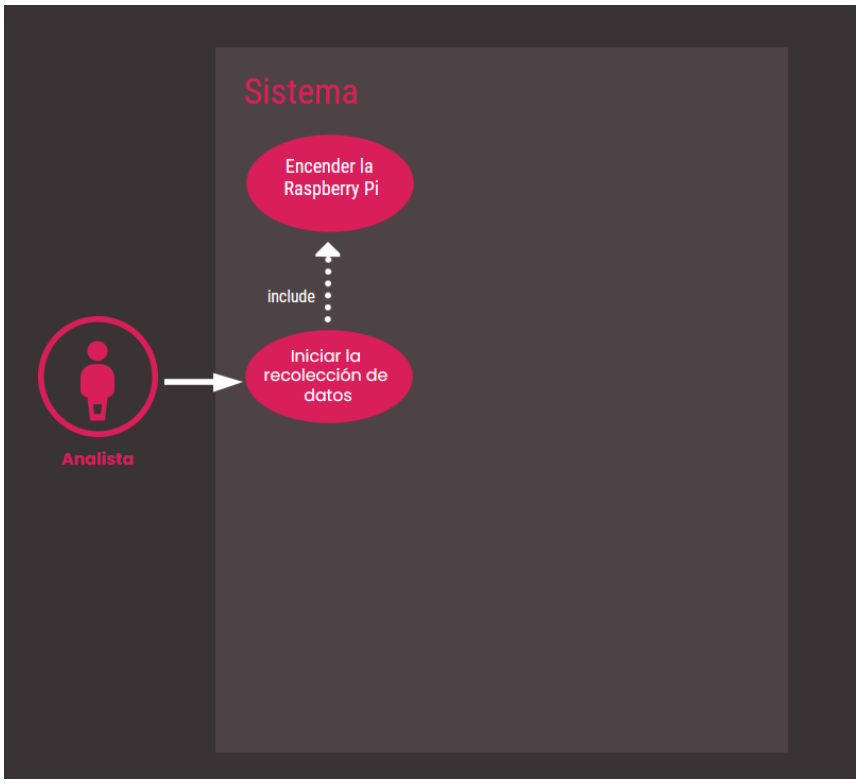


Figura 30. Diagrama caso de uso 3.

CDU-4	Finalizar la recolección de datos
Participantes	Analista
Descripción	El analista puede finalizar la recolección de datos en la base de datos.
Condición anterior	Se ha iniciado la recolección de datos.
Flujo natural	1-Desconectar la Raspberry Pi de la corriente eléctrica.
Flujo alternativo	No existe flujo alternativo
Condición posterior	Ninguna
Observaciones	Ninguna

Tabla 11. CDU-4 Finalizar la recolección de datos.

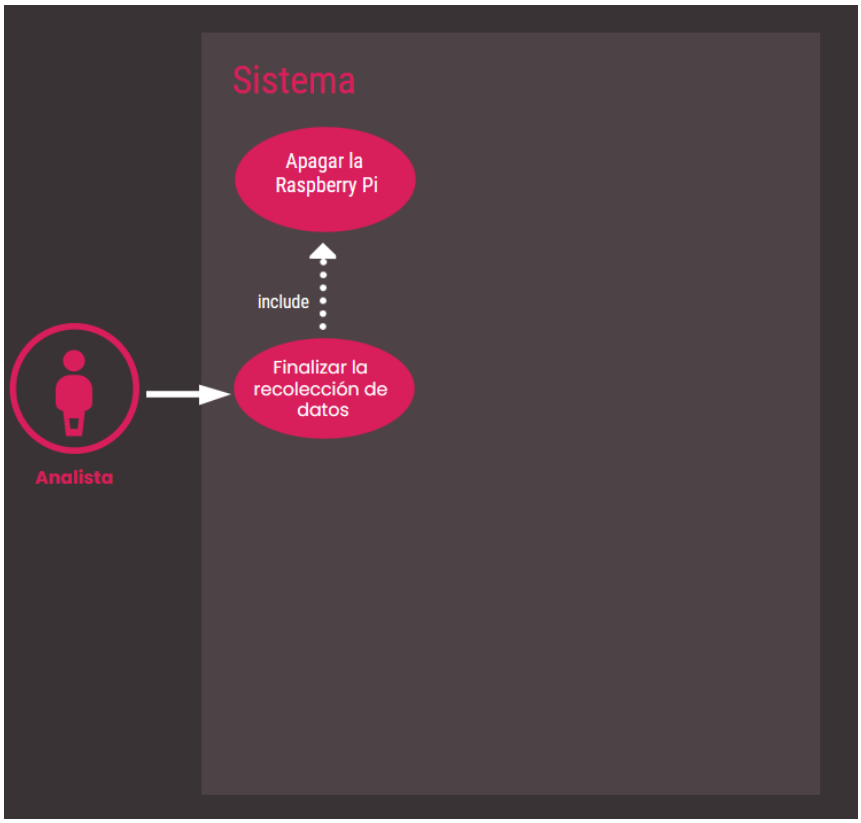


Figura 31. Diagrama caso de uso 4.

CDU-5	Editar la base de datos
Participantes	Administrador
Descripción	El administrador del sistema puede editar los datos contenidos en la base de datos.
Condición anterior	Haber creado un proyecto de Python en la Raspberry para la recolección de datos en la base de datos MongoDB.
Flujo natural	1-Iniciar la sesión en Raspberry Pi mediante ssh. 2-Acceder al proyecto de Python y editar la recolección de datos en MongoDB.
Flujo alternativo	1-Cambiar el proyecto Python alojado en GitHub. 2-Realizar una actualización del archivo de GitHub mediante la Raspberry.
Condición posterior	Reiniciar la Raspberry.
Observaciones	Ninguna

Tabla 12. CDU-5 Editar la base de datos.

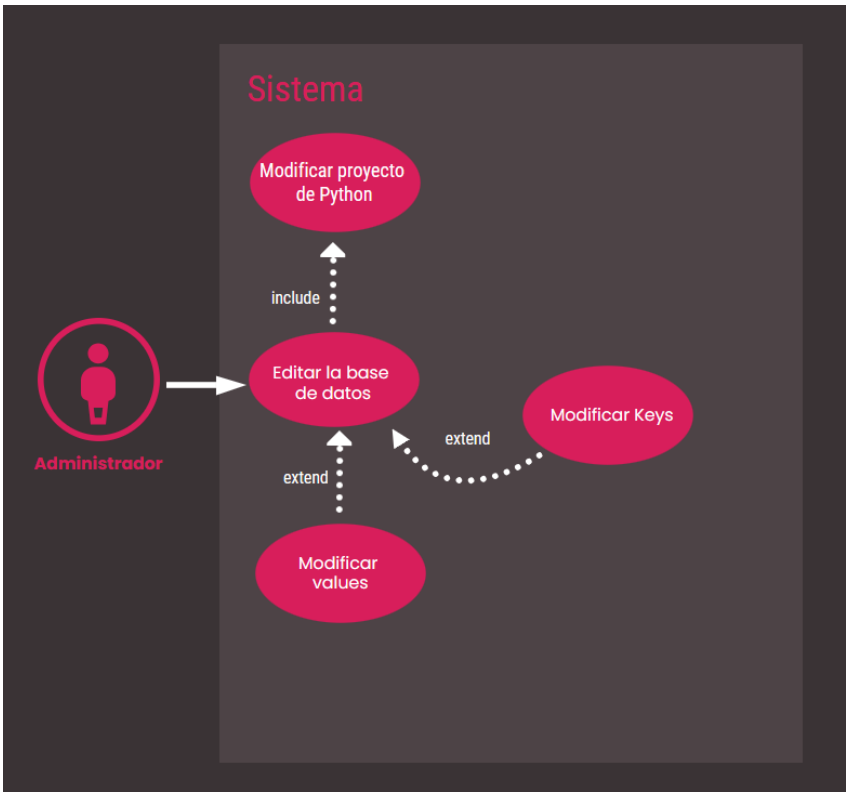


Figura 32. Diagrama caso de uso 5.

CDU-6	Editar la dashboard
Participantes	Administrador
Descripción	El administrador del sistema puede editar la dashboard creada en la plataforma IoT.
Condición anterior	Se ha creado una dashboard en la plataforma IoT (Home Assistant).
Flujo natural	1-Iniciar la sesión en Home Assistant. 2-Realizar cambios en la dashboard.
Flujo alternativo	No existe flujo alternativo
Condición posterior	Reiniciar Home Assistant.
Observaciones	Ninguna

Tabla 13. CDU-6 Editar la dashboard.

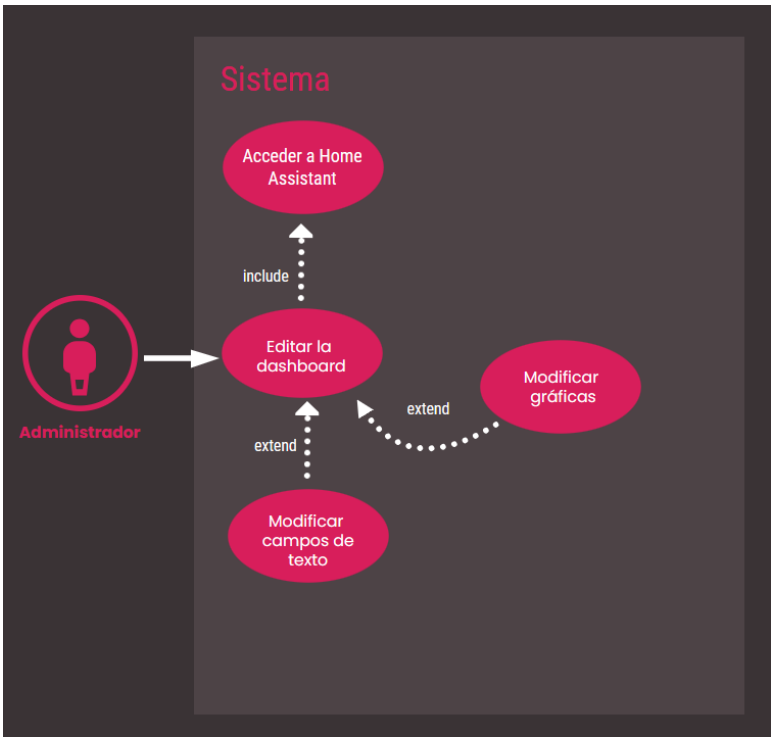


Figura 33. Diagrama caso de uso 6.

3. Implementación.

En este apartado de la memoria del TFM está dedicada a abordar el desarrollo del software que posteriormente se cargará tanto en las placas de desarrollo como en la Raspberry Pi.

El objetivo de nuestro proyecto era diseñar y construir un prototipo de sistema IoT funcional y completo capaz de monitorizar la calidad del aire en interiores, para posteriormente enviar los datos a un sumidero, donde se procesarían y se insertarían en la base de datos. Como se ha podido ver en secciones anteriores, ya se dispone del hardware necesario para cumplir nuestros objetivos, sin embargo, aún hace falta el software que proporcione las funcionalidades necesarias para tal propósito.

A continuación, se mostrará cómo se ha programado los microcontroladores escogidos para este proyecto. Para ello, se explicará el entorno de desarrollo utilizado para su programación y las ventajas respecto a otros entornos de desarrollo. Además, en esta sección, también se comentará el software escogido para la impresión en 3D de los soportes diseñados.

3.1 Entorno de desarrollo.

Como se ha comentado en apartados anteriores, en este proyecto, se ha utilizado placas de desarrollo ESP8266 para los sensores inerciales MPU6050 y la

ESP32 para el sensor de gas SEN0159. Este tipo de microcontroladores permite trabajar en diferentes entornos de desarrollo según nuestras necesidades. Debido a la envergadura de este proyecto se ha optado por la IDE (Integrated Development Environment) de Arduino ya que ofrece las funcionalidades básicas que necesitamos. La IDE de Arduino es un entorno de desarrollo de código abierto que destaca por:

- Lenguaje simple, basado en C/C++.
- Permite programar directamente sobre el hardware.
- Existe multitud de ejemplos de programación y tutoriales en Internet convirtiendo esta IDE en una de los más fáciles de usar.
- Dispone de librerías para prácticamente cualquier componente externo que se quiere acoplar.

El IDE de Arduino es un entorno de desarrollo que puede ser configurado para su utilización en placas de desarrollo tipo ESP. En los apartados 6.1 y 6.2 de este TFM se explica la configuración de esta IDE para su funcionamiento tanto en el microcontrolador ESP8266 como en el microcontrolador ESP32.

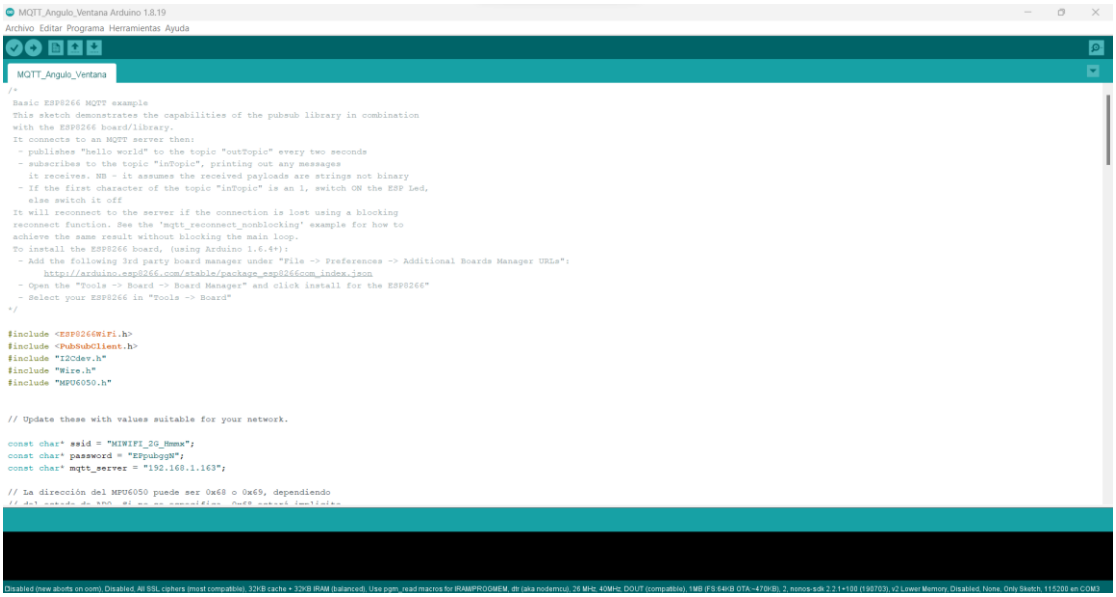


Figura 34. IDE de Arduino.

Como se ha comentado anteriormente, la IDE de Arduino permite introducir una gran variedad de librerías que complementa nuestro software añadiendo nuevas funcionalidades. En este proyecto se ha utilizado las siguientes librerías:

- **ESP8266Wifi.h y Wifi.h**: Librerías que permite tanto a la placa ESP8266 como a la placa ESP32 conectarse a la red Wi-fi.
- **PubSubClient.h**: Permite que la placa se comporte como un cliente MQTT, él cuál puede publicar y/o subscribirse a uno o varios tópics.

- **I2Cdev.h y Wire.h**: Permiten la utilización del protocolo de comunicación I2C. Este protocolo permite controlar varios dispositivos conectados sólo a dos hilos. Estos dos hilos corresponden a la línea SDA, encargada de transmitir los datos, y la línea SCL, encargada de enviar los sincronismos del reloj.
- **MPU6050.h**: Librería que permite obtener los valores de la aceleración y la velocidad angular que nos proporciona el sensor de inercia MPU6050.

El esquema general que siguen todos los programas de Arduino está formado por dos partes. Estas partes son las siguientes:

- **Setup**: Sólo se ejecuta una vez al iniciar el programa. En ella se declara las variables y se configura los pines.
- **Loop**: Esta función incluye todas las acciones que va a realizar el programa. A diferencia de la otra función, esta se ejecuta de forma cíclica durante todo el periodo de tiempo que el programa está funcionando.

Para el desarrollo del software de la Raspberry Pi se ha utilizado como entorno de desarrollo PyCharm. PyCharm es una IDE que permite crear programas en el lenguaje de programación Python, lenguaje elegido para este proyecto alojado en la Raspberry Pi. En nuestro caso, se ha utilizado la versión gratuita denominada PyCharm Community. Se ha utiliza esta IDE debido a que se trata de un entorno de desarrollo muy completo que permite editar, depurar e importar librería en Python de forma muy cómoda. En el apartado 6.3 de este TFM se explica cómo importar las librerías que necesitaremos para el desarrollo de nuestro proyecto en PyCharm.

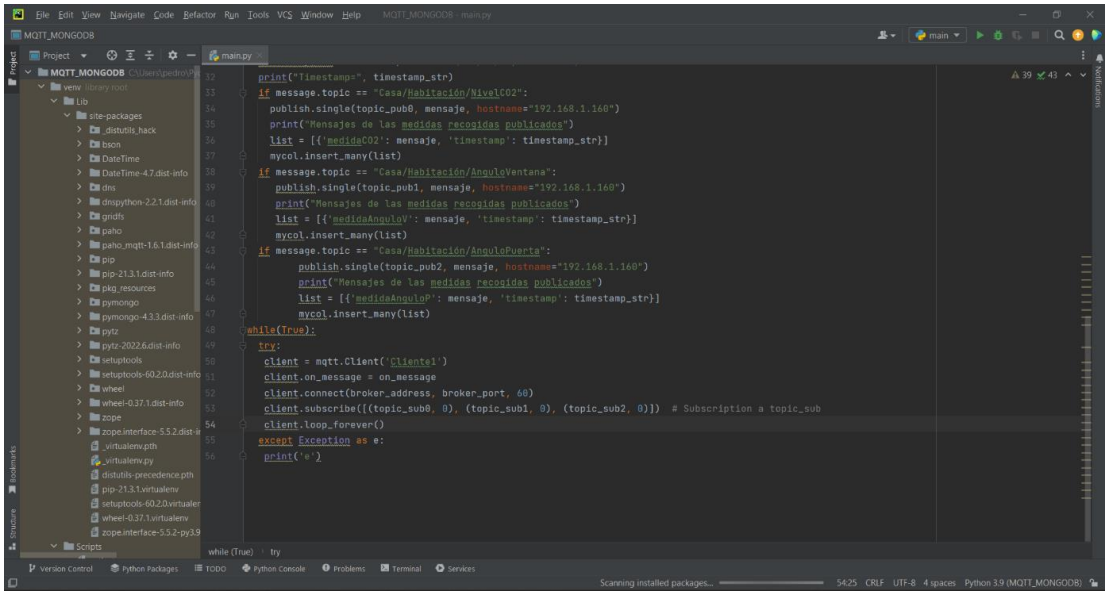


Figura 35. IDE de PyCharm.

1.1 Sumidero de datos.

El sumidero de datos se corresponde a la parte del sistema encargada de recoger los datos provenientes de las placas de desarrollo, en nuestro caso, el ángulo de apertura proporcionado por el MPU6050 y el nivel de CO_2 proporcionado por el sensor de gas. Esta función de sumidero de datos se realiza en la Raspberry Pi.

1.1.1 MQTT.

La comunicación entre los dispositivos se realiza mediante la conexión a la red Wi-fi y el protocolo de comunicación MQTT. MQTT [24] (Message Queing Telemetry Transport) tiene gran popularidad entre los dispositivos IoT por su ligereza y sencillez.

MQTT está basado en TCP/IP como base para la comunicación. MQTT funciona como servicio de mensajería publicador/suscriptor (pub-sub). Para filtrar los mensajes que son enviados a cada cliente los mensajes se disponen en topics organizados jerárquicamiente. Un cliente puede publicar un mensaje en un determinado topic. Otros clientes pueden suscribirse a este topic, y el bróker le hará llegar los mensajes suscritos.

Los clientes inician una conexión TCP/IP con el bróker, el cual mantiene un registro de los clientes conectados. Esta conexión se mantiene abierta hasta que el cliente la finaliza. Por defecto, MQTT emplea el puerto 1883 y el 8883 cuando funciona sobre TLS. Para ello el cliente envía un mensaje CONNECT que contiene la información necesaria (nombre de usuario, contraseña, client-id…). El bróker responde con un mensaje CONNACK, que contiene el resultado de la conexión (aceptada, rechazada, etc).

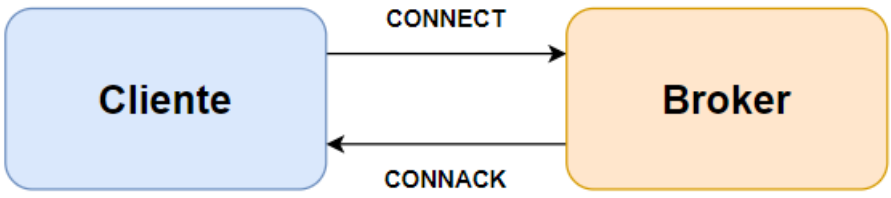


Figura 36. Proceso de conexión Cliente-Broker.

Para enviar el mensaje el cliente emplea mensajes PUBLISH, que contiene el topic y el payload.

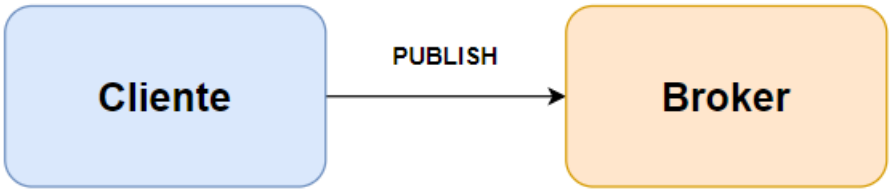


Figura 37. Proceso PUBLISH.

Para suscribirse y desuscribirse se emplean mensajes SUBSCRIBE y UNSUSCRIBE, que el servidor responde con SUBACK y UNSUBACK.

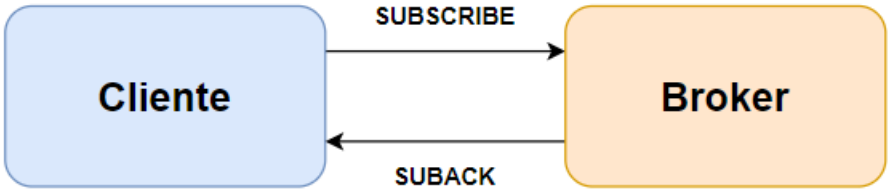


Figura 38. Proceso SUBSCRIBE.

- **Topic**

Los topics es un filtro que aplican los brokers MQTT a los mensajes que son recibidos desde los publicadores. El topic [25] consiste en una cadena de texto UTF-8 con una longitud máxima de 65536 caracteres. En el topic se distingue entre minúscula y mayúscula.

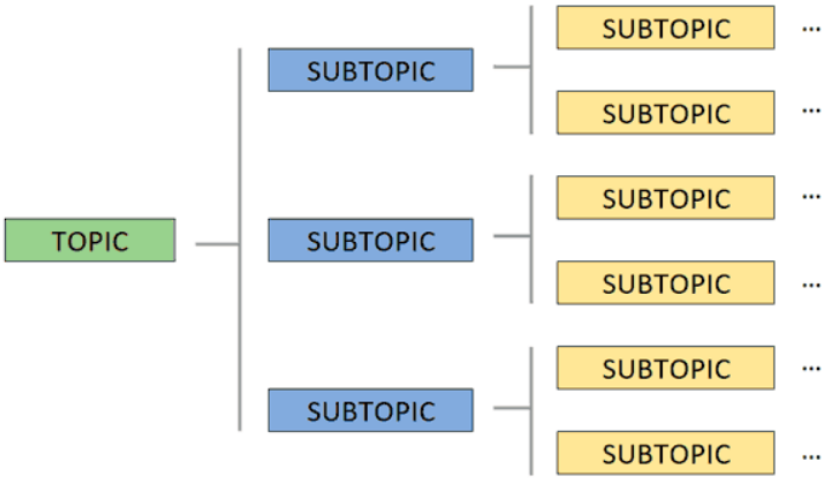


Figura 39. Niveles de los topics.

Los topics están formados por uno o más niveles separados entre sí por una barra inclinada. Cada nivel debe está formado por uno o más caracteres. En nuestro caso, para el envío de los datos desde las diferentes placas de desarrollo a la Raspberry Pi se ha utilizado topics con 3 niveles:

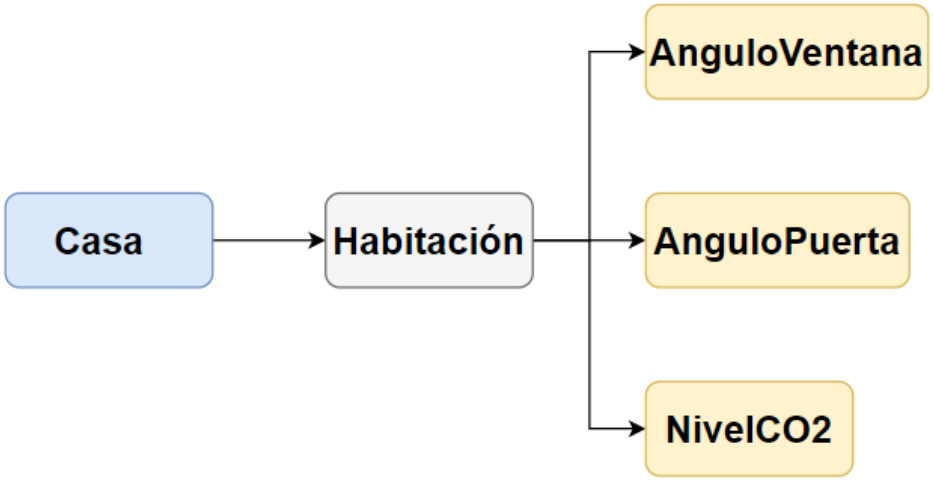


Figura 40. Topics de las placas de desarrollo.

Sin embargo, para el envío de estos datos desde el sumidero (Raspberry Pi) a la plataforma IoT se ha utilizado topics de dos niveles:

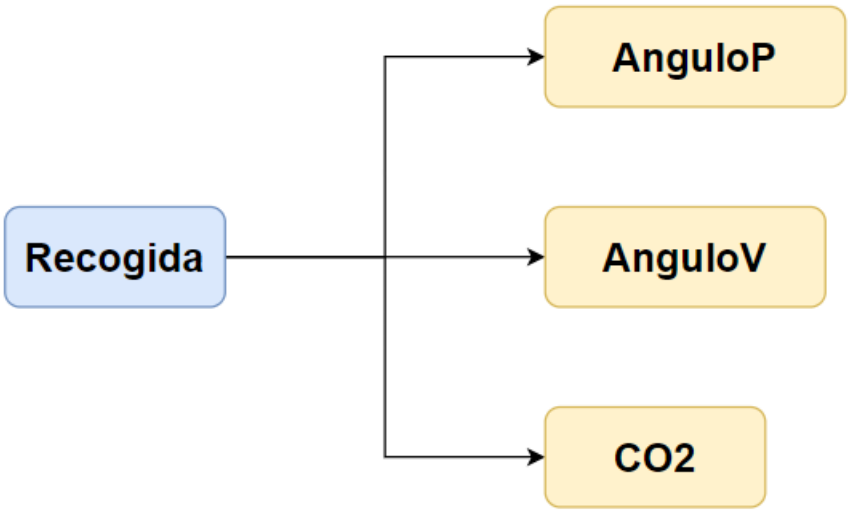


Figura 41. Topics de la Raspberry Pi.

Por tanto, para la comunicación entre las placas ESP8266 y la Raspberry Pi las suscripciones y publicaciones se realizan en los topics:

- "Casa/Habitación/AnguloPuerta".
- "Casa/Habitación/AnguloVentana".
- "Casa/Habitación/NivelCO2".

Mientras, para la comunicación entre la Raspberry Pi y la plataforma IoT las suscripciones y publicaciones se realizan en los topics:

- "Recogida/AnguloP".
- "Recogida/AnguloV".
- "Recogida/CO2".

- **Calidad de servicio (QoS)**

MQTT dispone de un mecanismo de calidad del servicio QoS, entendido como la forma de gestionar la robustez del envío de mensajes al cliente ante fallos (por ejemplo, de conectividad).

MQTT tiene tres niveles QoS posibles:

- **QoS 0 unacknowledged (at most one)**: El mensaje se envía una única vez. En caso de fallo por lo que puede que alguno no se entregue.
- **QoS 1 unacknowledged (at least one)**: El mensaje se envía hasta que se garantiza la entrega. En caso de fallo, el suscriptor puede recibir algún mensaje duplicado.
- **QoS 2 assured (exactly one)**: Se garantiza que cada mensaje se entrega al suscriptor, y únicamente una vez.

Usar un nivel u otro depende de las características y necesidades de fiabilidad de nuestro sistema. Un nivel de QoS superior requiere un mayor intercambio de mensajes de verificación con el cliente y, por tanto, mayor carga al sistema.

- **Estructura de un mensaje MQTT**

Uno de los componentes más importantes del protocolo MQTT es la definición y tipología de los mensajes, ya que son una de las bases de la agilidad en la que radica su fortaleza. Cada mensaje consta de 3 partes:

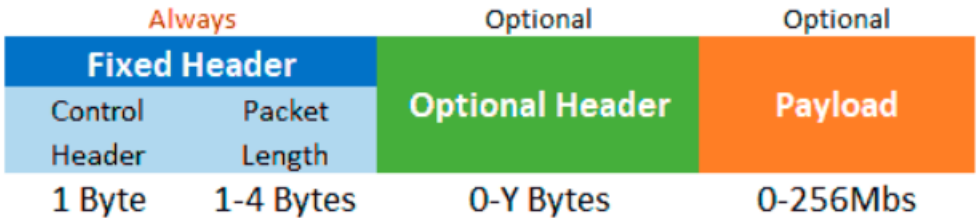


Figura 42. Estructura del mensaje MQTT.

- **Cabecera fija**: Ocupa 2 a 5 bytes, obligatorio. Consta de un código de control, que identifica el tipo de mensaje enviado, y de la longitud del mensaje. La longitud se codifica en 1 a 4 bytes, de los cuales se emplean los 7 primeros bits, y el último es un bit de continuidad.
- **Cabecera variable**: Opcional, contiene información adicional que es necesaria en ciertos mensajes o situaciones.
- **Payload**: Es el contenido real del mensaje. Puede tener un máximo de 256 Mb aunque en implementaciones reales el máximo es de 2 a 4 kB.

Los tipos de mensajes y códigos de control que se envían en el protocolo MQTT son los siguientes:

Message	Code
CONNECT	0x10
CONNACK	0x20
PUBLISH	0x30
PUBACK	0x40
PUBREC	0x50
PUBREL	0x60
PUBCOMP	0x70
SUSBSCRBE	0x80
SUBACK	0x90
UNSUSCRIBE	0xA0
UNSUBACK	0xB0
PINGREQ	0xC=
PINGRESP	0xD0
DISCONNECT	0xE0

Figura 43. Código de control que se envían en el MQTT.

- **Seguridad**

La seguridad siempre debe ser un factor importante a considerar en cualquier sistema de comunicación M2M. El protocolo MQTT dispone de distintas medidas de seguridad que podemos adoptar para proteger las comunicaciones.

Esto incluye transporte SSL/TLS y autentificación por usuario y contraseña o mediante certificado. Sin embargo, hay que tener en cuenta que muchos de los dispositivos IoT disponen de escasa capacidad, por lo que el SLL/TLS pude suponer una carga de proceso importante.

En muchos casos, la autentificación consiste en una contraseña y usuario que son enviados como texto plano. Por último, también es posible configurar el bróker para aceptar conexiones anónimas.

1.2 Base de datos.

En esta sección se explicará los distintos tipos de base de datos disponibles y se justificará la elección de MongoDB en nuestro proyecto.

1.2.1 Tipos de base de datos.

Algunos de los tipos de bases de datos que se utilizan actualmente son:

- **Base de datos relacionales o de lenguaje de consulta SQL (Structured Query Language)** [26]: Organiza los datos en filas y columnas, que en conjunto forman una tabla. Los datos normalmente se estructuran en varias tablas, que se pueden unir a través de una clave principal o una clave externa. SQL incluye la capacidad de contar, agregar, agrupar y también combinar consultas. SQL puede realizar funciones matemáticas básicas y subtotales y transformaciones lógicas. Los analistas pueden ordenar los resultados por fecha, nombre o cualquier columna. Las bases datos relacionales tienen varias ventajas en comparación con otros formatos de base de datos:
    - **Facilidad de uso**: En virtud de la vida útil de su producto, existe más comunidad en torno a las bases de datos relacionales, lo que perpetúa parcialmente su uso continuado.
    - **Redundancia reducida:** El propio modelo relacional reduce la redundancia de datos mediante un proceso conocido como normalización. Los procedimientos almacenados también ayudan a reducir el trabajo repetitivo.

- **Facilidad de copia de seguridad y recuperación de desastres**: Las bases de datos relacionales son transaccionales: garantizan que el estado de todo el sistema sea consistente en todo momento. La mayoría de bases de datos relacionales ofrecen opciones sencillas de exportación e importación, lo que hace que la copia de seguridad y restauración sean triviales.

Entre los gestores de base de datos relacionales más utilizados en la actualidad se cuentan Db2, Microsoft SQL Server, MySQL, PostgreSQL, Oracle o SQLite.

- **Base de datos no relacionales [27]:** También conocido como base de datos NoSQL. En lugar de la estructura tabular típica de una base de datos relacional, las bases de datos NoSQL albergan datos dentro de una estructura de datos, como un documento JSON. Este tipo de bases de datos tienen las siguientes ventajas respecto a sus competidores:
  - **Rentabilidad:** Es costoso mantener un sistema de gestión de base de datos relacionales ya que requiere compra de licencias, administradores de base de datos capacitados y hardware potente para escalar verticalmente. Las bases de datos NoSQL le permiten escalar rápidamente horizontalmente, asignando mejor los recursos para minimizar los costos.
  - **Flexibilidad:** Las bases de datos NoSQL pueden abordar grandes volúmenes de datos que cambian rápidamente, lo que las hace ideales para el desarrollo ágil, iteraciones rápidas e implementaciones de códigos frecuentes.
  - **Réplica:** la funcionalidad de réplica NoSQL copia y almacena datos en varios servidores. Esta réplica brinda confiabilidad a los datos, y garantiza el acceso durante tiempos de inactividad y protección frente a la pérdida de datos si los servidores se desconectan.
  - **Velocidad:** NoSQL permite un almacenamiento y un procesamiento más rápidos y ágiles para todos los usuarios.

Entre las bases de datos no relacionadas más utilizadas se encuentran Cassandra, MongoDB, CouchDB, SimpleDB o BigTable.

1.2.2 Selección de la base de datos.

En este proyecto se ha optado por utilizar la base de datos, de tipo NoSQL, MongoDB. La elección de esta base de datos es debido a que, entre otras cosas, permite la creación de índices para acelerar las búsquedas, lo que resulta de utilidad cuando el volumen de datos se incremente notablemente. Además, MongoDB guarda estructuras de datos BSON con un esquema dinámico, haciendo que la integración en ciertas aplicaciones sea más fácil y rápida.

En nuestro proyecto, se ha decidido que se utilizará tres entidades, una para cada sensor que hay en el sistema, las cuáles no se relacionan entre sí. En las siguientes tablas se mostrará las propiedades que deben tener para el correcto funcionamiento de nuestro sistema:

NivelCO2	Descripción	Tipo de variable
_id	Identificador único	ObjectId
medidaCO2	PPM de CO2	String
timestamp	Fecha y hora (Zona horaria de España)	String

Tabla 14. Colección de datos del sensor SEN0159.

AnguloVentana	Descripción	Tipo de variable
_id	Identificador único	ObjectId
medidaAnguloV	Valor del ángulo de apertura de la ventana	String
timestamp	Fecha y hora (Zona horaria de España)	String

Tabla 15. Colección de datos del sensor MPU6050 en la ventana.

AnguloPuerta	Descripción	Tipo de variable
_id	Identificador único	ObjectId
medidaAnguloP	Valor del ángulo de apertura de la puerta	String
timestamp	Fecha y hora (Zona horaria de España)	String

Tabla 16. Colección de datos del sensor MPU6050 en la puerta.

1.3 Prototipo de soportes 3D.

Como software de impresión 3D para la impresora se ha utilizado el Ultimate Cura debido a su simplicidad en el uso. Además, Ultimate Cura es un software de código abierto y ampliamente utilizado en todo el mundo por lo que está en constante evolución.

El material utilizado para la fabricación de los soportes es PLA (ácido poliláctico) por ser altamente resistente a la humedad, característica beneficiosa para este proyecto.

1.4 Plataforma IoT para la visualización de datos.

En este apartado, en primer lugar, se explicará qué es una plataforma IoT y sus principales componentes. A continuación, se expondrán la plataforma IoT más importantes del mercado actualmente. Finalmente, se justificará la elección de la plataforma IoT elegida para este proyecto.

1.4.1 Definición y principales componentes.

La plataforma IoT reúne un conjunto de servicios que recopilan, almacenan, correlacionan, analizan y explotan datos. Se refiere, por tanto, al software de soporte que conecta todo el sistema IoT, facilitando la comunicación, el flujo de datos, la gestión de dispositivos y la funcionalidad de la aplicación. La plataforma conecta dispositivos a una nube con opciones de conectividad flexibles.

Para los desarrolladores, una plataforma IoT proporciona un conjunto de características listas para usar que aceleran el desarrollo de aplicaciones para dispositivos conectados. Para que una plataforma IoT pueda funcionar de forma correcta es necesario que tenga implementado ciertos componentes. A continuación, se detalla cada uno de ellos:

- **Conectividad:** Es la parte central de cualquier aplicación IoT en la industria, pues garantiza que exista una adecuada interacción entre los diversos dispositivos y que haya una transmisión exacta de los datos. De esta forma, tanto los dispositivos en el campo como los que se encuentran en la nube se puedan conectar de forma idónea, con el objetivo de que se logre almacenar y analizar la información con mayor facilidad.
- **Administración de conectores:** Es fundamental que se tenga un control y supervisión de todos los dispositivos conectados tanto en terreno como en la nube. Esta administración es clave para garantizar que todas las aplicaciones IoT en la industria funcionen de manera correcta.
- **Sistema de gestión centralizado:** A través de la integración de un sistema de módulo de gestión, es posible recopilar información básica del estado de los diferentes equipos que han sido desplegados. Además, a través de dicho módulo se pueden llevar a cabo de forma remota las debidas actualizaciones del software.
- **Entorno integrado para las aplicaciones:** Desde el entorno de la plataforma IoT, los usuarios podrán ejecutar sus sistemas haciendo uso de interfaces

gráficas bastantes intuitivas, en las que podrán mover y arrastrar fácilmente cada uno de los elementos. Esto es posible gracias a que dichos entornos vienen preparados con funcionalidades de bajo código (low code) que permite que usuarios sin conocimientos de programación puedan diseñar sus propios interfaces y aplicaciones.

- **Distribución de aplicaciones y envío de comandos:** Todas las aplicaciones, actualizaciones de firmware o parámetros de los dispositivos se despliegan de manera remota cumpliendo los protocolos de seguridad.
- **Bases de datos:** Las bases de datos son indispensables y pueden estar on premise o bien en la nube. A medida que crece el número de dispositivos que se conectan a ellas, van escalando para poder almacenar todos los registros correctamente.
- **Análisis y visualización de datos:** Los datos de por sí, no representan gran utilidad si no se analizan en profundidad. Por eso, es necesario visualizar de forma detallada esta información que suele presentarse a través de gráficos o indicadores. Generalmente, la gran mayoría de plataformas tienen incluidas herramientas de transformación y visualización. Incluso permiten que se puedan integrar herramientas de terceros.
- **Ciberseguridad:** Una plataforma IoT industrial debe ser, por un lado, robusta y por otro, garantizar que toda la información alojada se encuentre segura. Es fundamental que diferentes aspectos como la aplicación de autenticación, el cifrado de datos, la restricción de servicios de red y la anulación de puertos, estén en el foco de la ciberseguridad.

1.4.2 Principales plataformas IoT del mercado.

Algunas de las principales plataformas IoT del mercado en la actualidad son las siguientes:

- **Google Cloud IoT:** Se trata de una plataforma IoT Open Source. Esta plataforma incluye dos ingredientes importantes que dan soluciones al ecosistema IoT, como son BIG Query y Google Cloud Data. Esta combinación ayuda a analizar los datos de forma precisa y nos proporciona resultados relevantes. Algunos de sus beneficios son:
    1. Maneja grandes cantidades de datos.
    2. Proporciona información crucial.

3. Simplifica el proceso de trabajo en sistemas IoT.

- **IBM Watson:** Desarrollada por IBM, esta plataforma es un servicio completamente gestionado y alojado en cloud con funcionalidades para el registro de dispositivos, la conectividad, el control, la visualización rápida y el almacenamiento de datos. Para establecer las conexiones utiliza el protocolo MQTT. Algunos de sus beneficios son:
    1. Tablero simple y permite una mejor visualización de datos.
    2. Transmisión segura y almacenamiento de dispositivos.
    3. Minería de datos sobre la marcha.
- **Amazon Web Services**: Es una colección de servicios de computación en la nube pública que en conjunto forman una plataforma de computación en la nube. Se accede a esta plataforma a través de HTTP, utilizando protocolos REST y SOAP. Algunos de sus beneficios son:
    1. Distribución de aplicaciones a varias redes.
    2. Mantiene la durabilidad de la aplicación.
    3. Garantiza la disponibilidad de las aplicaciones.
- **Home Assitant**: Es una plataforma IoT Open Source que se codifica mediante el lenguaje de programación Python. Suele estar dedicada a sistemas domóticos. Para establecer las conexiones con los demás dispositivos permite utilizar el protocolo MQTT. Algunos de sus beneficios son:
    1. Es ideal para alojar en una Raspberry Pi y tener conectados todos los dispositivos desde una red local.
    2. Se pueden integrar APIs (como WeMo o Nest) y automatizar tareas a partir de IFTTT (“If This, Then That”).
    3. Tus datos no salen de tu casa, es decir, no dependes de servidores en la nube, o aplicaciones de terceros. Todo lo controlas desde tu servidor local.

3.5.3 Selección de la plataforma IoT.

Tras haber expuesto algunas de las principales plataformas IoT del mercado y teniendo en cuenta que en este proyecto se busca principalmente la visualización de los datos en tiempo real, se ha elegido Home Assistant.

4. Desarrollo del proyecto.

En este apartado de la memoria, tras haber comentado en secciones anteriores toda la tecnología que ha formado parte del desarrollo del sistema de monitorización de la calidad del aire en interiores, se explicará la localización elegida para la implementación de este sistema. Además, se describirán las diferentes iteraciones e incrementos que han ido marcando el desarrollo de este proyecto, siguiendo la metodología elegida en el apartado 1.7.3.

4.1 Espacio físico.

Debido a la naturaleza del propio trabajo, es necesario un espacio para configurar, desplegar y experimentar el sistema de monitorización. En este caso, el trabajo se ha llevado a cabo en una habitación de aproximadamente 12 m^2. En la siguiente figura se muestra el plano de la habitación:

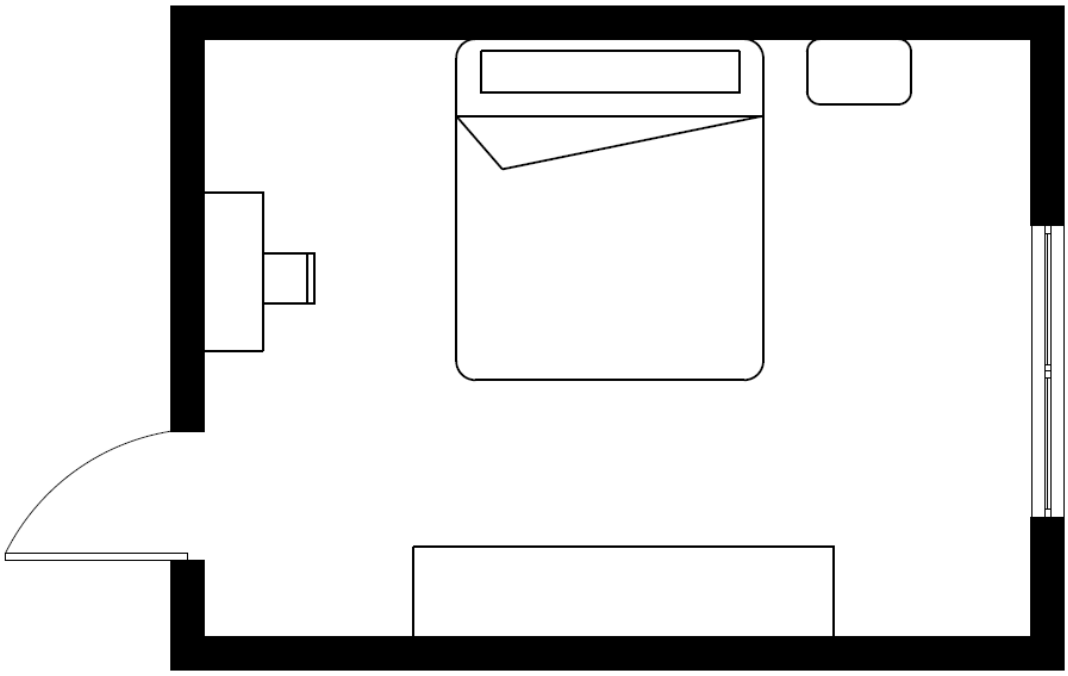


Figura 44. Plano de la habitación donde se implementa el sistema.

Los sensores MPU6050 se han colocado tanto en la ventana como en la puerta con el objetivo de obtener el ángulo de apertura. El sensor de gas SEN0159 se ha colocado encima del armario que hay entre la ventana y la puerta, en la zona intermedia de la habitación. De esta forma, se consigue una mayor precisión en la calibración del sensor debido a que se localiza en una zona donde la apertura, ya sea

de la ventana o la puerta, no infla los valores obtenidos por el sensor debido a la cercanía de estas.

4.2 Iteraciones.

En este apartado se irá describiendo los diferentes hitos conseguidos en el desarrollo del sistema que nos han llevado a la finalización de este.

- **1º Iteración: Familiarización con el sensor SEN0159.**

La primera iteración tiene como objetivo la familiarización con el sensor SEN0159, elegido por su mayor precisión en la medida del CO_2 respecto a otros sensores del mercado. Para ello, en esta iteración sólo disponemos de una placa ESP32-WROOM-32D, una protoboard, cables, el sensor de gas, la fuente de alimentación MB102 y resistencias para poder construir el divisor de tensión, imprescindible como ya se comentó en apartados anteriores. A partir de estos elementos, el prototipo construido nos mostrará los valores del nivel de CO_2 que hay en el ambiente a través del puerto serie de Arduino. En la figura 21 se puede observar el montaje realizado en la placaboard con los elementos nombrados.

En esta primera iteración se ha configurado el IDE de Arduino para su funcionamiento con la placa de desarrollo ESP32 como se muestra en el apartado 6.2 de esta memoria.

- **2º Iteración: Montaje del sensor MPU6050.**

En esta segunda iteración, se disponía del sensor MPU6050, cables y la placa de desarrollo NodeMCU Amica con el objetivo de obtener los valores del ángulo de apertura, ya sea de la puerta o la ventana. En este caso, estos valores también se mostrarán en el puerto serial de Arduino. En la figura 19 se muestra el montaje de los diferentes elementos en la protoboard.

El cálculo del ángulo se ha explicado en el apartado 2.3.2. Para la obtención correcta del ángulo de apertura, antes se debe calibrar el sensor MPU6050. Esta calibración se ha realizado mediante el sketch MPU6050_Calibration realizado por Luis Ródenas. Este programa hace la calibración del sensor y nos proporciona unos offsets que se utilizarán para la autocalibración del sensor. Para el correcto funcionamiento de este sketch, el sensor debe estar colocado de forma que el eje Z coincida con la gravedad. Además, los offsets obtenidos depende de la posición del sensor, por lo tanto, es conveniente tener el sensor siempre en la misma posición.

Debido a esto, se decidió el diseño de un soporte para la protoboard que sustenta al MPU6050. De esta forma, el sensor siempre va a estar en la misma posición.

- **3º Iteración: Inclusión del protocolo MQTT.**

El objetivo de esta tercera iteración era incluir el protocolo MQTT para la comunicación entre las placas ESP8266, la placa ESP32 y la Raspberry Pi. Aquí entra en juego la Raspberry Pi como nodo sumidero de todos los datos obtenidos por los sensores. En primer lugar, se instaló el sistema operativo en la Raspberry Pi como se explica en la sección 6.2 de esta memoria. A continuación, se instaló Mosquitto en la Raspberry Pi. Mosquitto es un bróker MQTT que nos permitirá suscribirnos, desde la Raspberry Pi, a los topics publicados por las placas ESP8266 y la placa ESP32.

Para la inclusión del protocolo MQTT en los códigos realizados, en la IDE de Arduino, en las iteraciones anteriores, se ha utilizado la librería PubSubClient.h. Esta librería permite publicar y suscribirse a topics desde las placas ESP. La elección de esta librería es por su compatibilidad con las placas ESP y su sencillez, ya que viene con numerosos ejemplos a los que nos podemos apoyar.

- **4º Iteración: Integración de MongoDB.**

La cuarta iteración tiene como objetivo la integración de la base de datos elegida, MongoDB, en nuestro sistema. Con esto, conseguiremos que nuestros datos persistan en el tiempo. Para la integración de MongoDB en nuestro sistema, se ha tenido que instalar PyCharm Community, IDE que nos permitirá realizar programas en lenguaje Python. La inclusión de las diferentes librerías que necesitamos se ha realizado según se comenta en el apartado 6.3 de esta memoria.

El script realizado en Python comunica a la Raspberry Pi con las diferentes placas de desarrollo mediante el protocolo MQTT. La Raspberry Pi se suscribe a los diferentes topics publicados por las distintas placas con el objetivo de recibir los datos de los diferentes sensores. Según el topic al que se suscribe, el programa almacena los datos recibidos en una de las tres entidades creadas en MongoDB. Estas tres entidades son AnguloVentana, AnguloPuerta y NivelCO2.

Para facilitar la labor del analista, se ha automatizado el inicio y finalización de la recolección de datos en MongoDB. El script creado en Python para la recolección de datos se ejecutará al conectar la Raspberry Pi a la corriente eléctrica. El script seguirá corriendo hasta que la Raspberry Pi se desconecte de la corriente eléctrica. Esta automatización se ha realizado de la siguiente manera:

1º. **Creación de un entorno virtual e instalación de las librerías utilizadas en el script de Python:** Una vez que se ha alojado el script creado en PyCharm Community en la Raspberry, se ha creado un entorno virtual en esta. Los entornos virtuales en Python nos proporcionan un entorno de desarrollo aislado para proyectos individuales, los que nos permite instalar sólo los paquetes necesarios para el proyecto específico. Para la creación de este entorno virtual se ha tenido que instalar virtualenv mediante la siguiente línea de código:

```
sudo apt install dh-virtualenv
```

Tras realizar esta instalación, se ha instalado las librerías utilizadas en el script de Python, explicadas en el apartado 6.3 de la memoria. Para la instalación de estas librerías en el entorno virtual, primero se ha tenido que activar el entorno virtual mediante la siguiente línea de código:

```
source /home/pi/MongoDB/env/bin/actívate
```

Para la instalación de las librerías se ha ejecutado la siguiente línea de código:

```
pip install librería
```

En librería pondríamos paho, pytz, datetime o pymongo según la librería a instalar.

2º. **Creación de un script de shell**: El siguiente paso para esta automatización es la creación de un script de shell que permita activar el entorno virtual creado y ejecutar el script de Python. La creación del archivo MongoDB.sh donde escribiremos este script se realiza mediante la siguiente línea de código:

```
touch MongoDB.sh
```

En el archivo creado, para la activación del entorno virtual y la ejecución del script de Python, se escribirán las siguientes líneas de código:

```
#!/bin/bash
source /home/pi/MongoDB/env/bin/activate
python3.9 /home/pi/MongoDB/MongoDB.py
```

3º. **Utilizar Crontab para la ejecución del script de shell al arrancar la Raspberry Pi**: Crontab es una herramienta que nos permite programar que se ejecute ciertas tareas en ciertas condiciones. En nuestro caso, utilizaremos esta herramienta para ejecutar el script creado en el paso anterior al arrancar la Raspberry Pi. Para ello, primero debemos ejecutar la siguiente línea de código:

```
sudo crontab -e
```

Esta línea de código nos dará acceso a Crontab. Una vez aquí, escribiremos la siguiente línea de código al final del archivo:

```
@reboot bash /home/pi/MongoDB.sh
```

Con esta línea de código, el código escrito en el archivo shell se ejecutará al iniciar la Raspberry Pi, es decir, se iniciará la recolección de datos en la base de datos MongoDB.

- **5º Iteración: Visualización de los datos a través de una plataforma IoT.**

El objetivo de esta quinta iteración es la integración de la plataforma IoT en nuestro sistema con el objetivo de visualizar los datos en tiempo real. Para ello, primero se ha tenido que instalar Home Assistant, la plataforma IoT elegida, en la Raspberry Pi. En el apartado 6.4 se explica con detalle el proceso de instalación. Una vez se ha tenido acceso a Home Assistant, se ha modificado el script de Python realizado en la iteración anterior. En esta modificación, hemos incluido la comunicación mediante MQTT entre la Raspberry Pi y la plataforma IoT. En esta iteración, se le añade código al script. En el código añadido, la Raspberry Pi publica topics según los datos recibidos de los sensores. Ahora, Home Assistant deber suscribirse a estos topics para recibir los datos y así poder visualizarlos en una dashboard. Para suscribirse a estos topic, se ha debido instalar una integración de MQTT que permita a la plataforma comunicarse mediante MQTT. Una vez instalada, se ha añadido unas líneas de código en el archivo Configuration.Yaml para suscribirse a los topics publicados por la Raspberry Pi.

```
mqtt:
  sensor:
    - name: "Ángulo de la puerta"
      state_topic: "Recogida/AnguloP"
      unit_of_measurement: "º"
    - name: "Ángulo de la ventana"
      state_topic: "Recogida/AnguloV"
      unit_of_measurement: "º"
    - name: "Nivel de CO2"
      state_topic: "Recogida/CO2"
      unit_of_measurement: "ppm"
```

Figura 45. Código añadido al archivo Configuration.Yaml

Los datos recibidos se muestran en una dashboard. La construcción de esta dashboard se ha dividido en dos partes:

1. En la parte superior, primero nos encontramos con una gráfica que nos indica el nivel de CO_2 en ppm. Además, la flecha de la barra indicadora se situará en uno de los tres colores según este nivel. La flecha se situará en la zona verde cuando el nivel de CO_2 está entre 400 ppm y 800 ppm, indicando que la calidad del aire en interior es buena. La flecha se situará en la zona amarilla cuando el nivel de CO_2 está entre 800 ppm y 1200 ppm, indicando que la calidad del aire

en interior es baja. Por último, la flecha se situará en la zona roja cuando el nivel de CO_2 está por encima de los 1200 ppm, indicando que la calidad del aire es mala. Además de esta gráfica, en la parte superior izquierda encontramos un indicador de valores, que nos mostrará en tiempo real los valores obtenidos por los sensores, es decir, el ángulo de la puerta, el ángulo de la ventana y el nivel de CO_2.

2. En la parte inferior, se encuentran dos gráficas donde se muestra el histórico de medidas. La gráfica situada en la izquierda muestra el histórico tanto del ángulo de apertura de la ventana como el ángulo de apertura de la puerta. La gráfica situada en la derecha muestra el histórico del nivel de CO_2 que hay en la habitación.

Figura 46. Dashboard.

5. Validación y evaluación.

En este apartado del TFM se presenta los casos de estudio que se han llevado a cabo para comprobar el buen funcionamiento del sistema prototipo de monitorización de la calidad de aire en interiores y el cumplimiento de los objetivos requeridos.

5.1 Casos de estudio.

La experimentación se va a basar en la observación del progreso del nivel de CO_2 en diferentes condiciones con el paso del tiempo:

- Ventana y puerta abiertas 83º con una persona en el interior acostada.
- Ventana abierta 83º y puerta completamente cerrada (0º) con una persona en el interior acostada.

- Ventana y puerta completamente cerrada con una persona en el interior acostada.
- Ventana y puerta abiertas 83º con una persona corriendo en el interior.
- Puerta abierta 83º y ventana completamente cerrada con una persona corriendo en el interior.
- Ventana y puerta completamente cerradas con una persona corriendo en el interior.

5.1.1 Primer caso de estudio.

En este primer caso, se realiza una sesión de aproximadamente 1 hora en el que una persona permanece acostada en la habitación mientras la puerta y la ventana permanecen abiertas a 83 grados.

Figura 47. Dashboard en el primer caso de estudio.

En la imagen se puede observar que esta sesión empieza a las siete y veinticinco de la tarde aproximadamente y termina las ocho y veinticinco. Se puede ver, en la gráfica del nivel de CO_2, que se ha ido cogiendo muestras cada segundo. Se muestra una variabilidad constante en el nivel de CO_2 entre valores de 403-440 ppm con algunos picos que alcanzan los 500 ppm. Para disminuir esta variabilidad se ha decidido representar, además de los valores brutos, la media de cada 5 muestras como se observa en la siguiente figura:

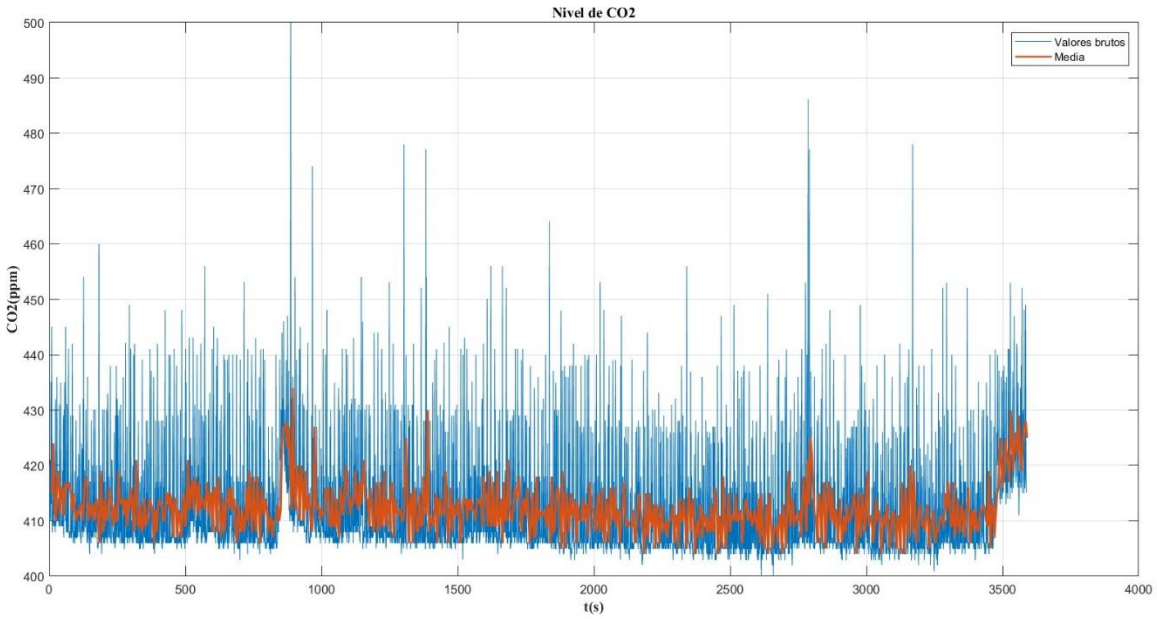


Figura 48. Nivel de CO2 a lo largo del tiempo (Caso 1).

Si nos fijamos en la línea roja, podemos observar que el máximo se produce aproximadamente a los 900 segundos y tiene un valor de 434 ppm. Además, se puede ver que el nivel de CO_2 permanece casi constante a 410 ppm durante la hora que ha durado la sesión. Con la ventana y la puerta abierta nos movemos en valores bajos del nivel de CO_2 ya que, en el exterior, el nivel de CO_2 es, aproximadamente, a 400 ppm.

5.1.2 Segundo caso de estudio.

En este segundo caso, se realiza una sesión de aproximadamente 1 hora en el que una persona permanece acostada en el interior de la habitación mientras la puerta permanece cerrada y la ventana abierta 83 grados.

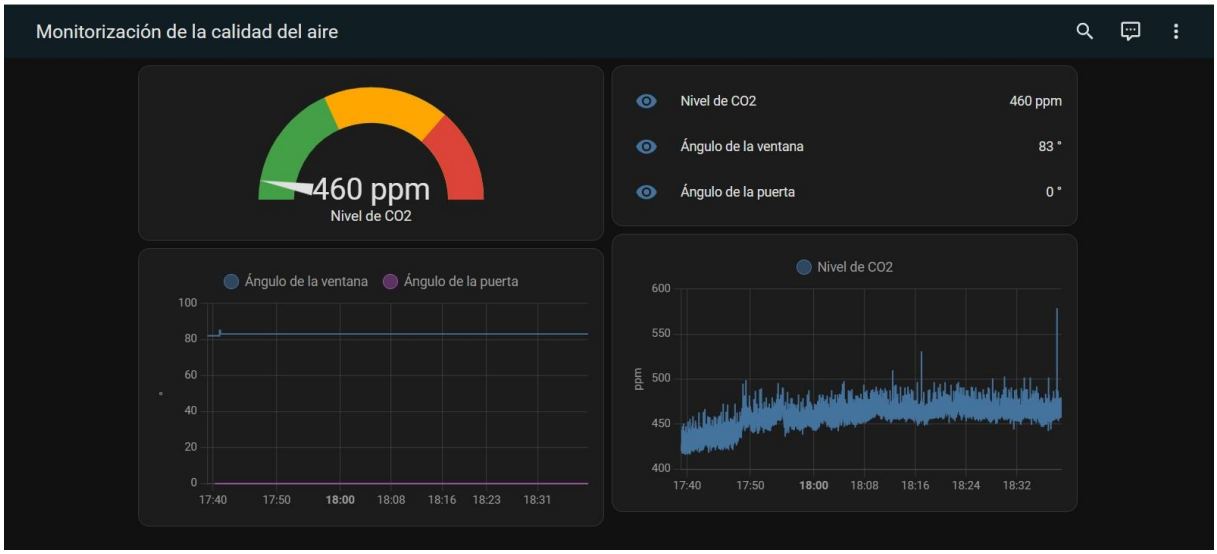


Figura 49. Dashboard en el segundo caso de estudio.

En la figura 47 se puede observar que esta segunda sesión empezó a las seis menos veinte de la tarde y acabó a las tiene menos veinte. En este caso, el nivel de CO_2 empieza oscilando, aproximadamente, entre valores de 420-440 ppm y acaba oscilando entre valores de 450-490 ppm. Como se ha explicado anteriormente, debido a la frecuencia de obtención de muestras, se ha decidido realizar una gráfica en el que, además de los valores brutos, se muestre la media de cada 5 muestras.

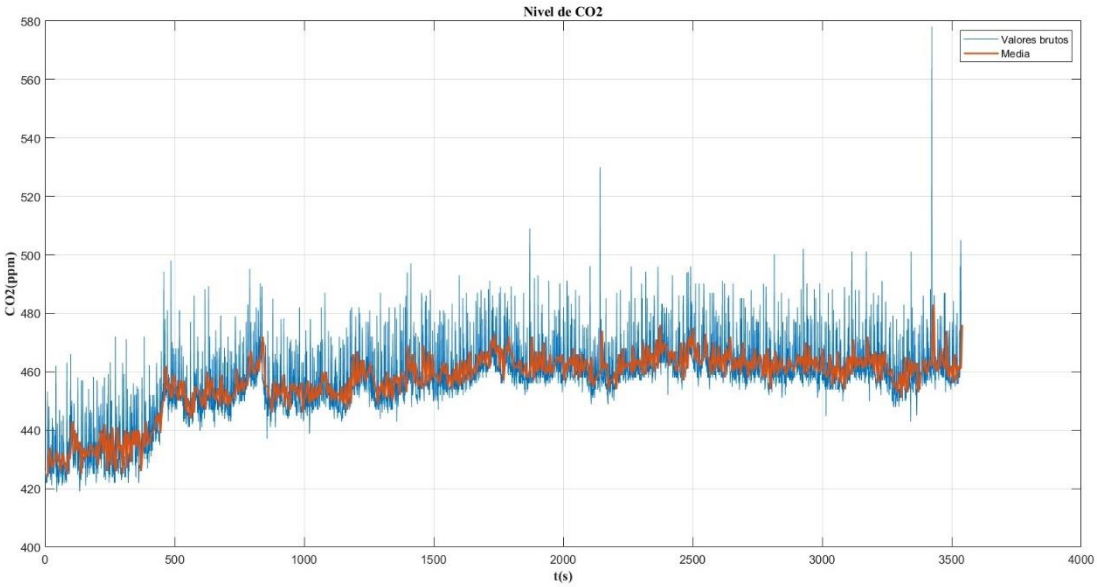


Figura 50. Nivel de CO2 a lo largo del tiempo (Caso 2).

En la figura 48, se puede observar a través de la línea roja que el nivel de CO_2 aumenta hasta que llega a un valor aproximado a 460 ppm, donde se estabiliza. El máximo se produce aproximadamente a los 3400 segundos y tiene un valor de 483 ppm.

5.1.3 Tercer caso de estudio.

En este tercer caso, se ha realizado una sesión de una hora aproximadamente en el que una persona permanece acostada mientras tanto la puerta como la ventana están completamente cerradas.

Figura 51. Dashboard en el tercer caso de estudio.

En esta imagen se puede observar como la sesión empieza a las 17:05 y termina a las 18:05 aproximadamente. En este caso, el nivel de CO_2 tiene una tendencia ascendente llegando a superar los 700 ppm llegado la hora. Es decir, una persona acostada durante 1 hora con la puerta y la ventana cerrada llega a unos niveles de CO_2 en el que la calidad del aire es baja.

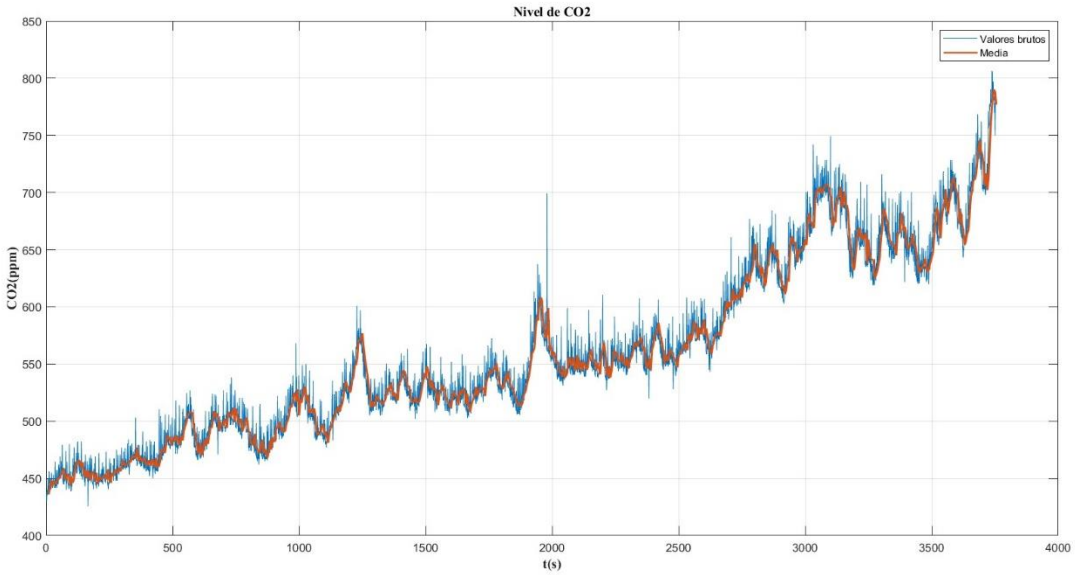


Figura 52. Nivel de CO2 a lo largo del tiempo (Caso 3).

Aquí la línea roja tiene una tendencia ascedente. En este caso, el nivel de CO_2 no se estabiliza en un valor, como ocurría en los dos casos anteriores. Esto es debido a que, al estar la habitación completamente cerrada, no hay renovación aire y, por lo

tanto, los niveles de CO_2 aumentan a lo largo del tiempo. El máximo se produce 3755 segundo, es decir, a la hora aproximadamente y su valor es de 790 ppm.

5.1.4 Cuarto caso de estudio.

En este cuarto caso, se ha realizado una sesión de aproximadamente veinte minutos en el que una persona realiza ejercicio mientras la puerta y la ventana están completamente cerradas.

Figura 53. Dashboard en el cuarto caso de estudio.

En la figura 51 se observa como la sesión empieza aproximadamente a las 9:15 y termina, aproximadamente, a las 9:35. Se puede observar un ascenso del nivel de CO2 llegando a superar niveles de 800 ppm en 20 minutos.

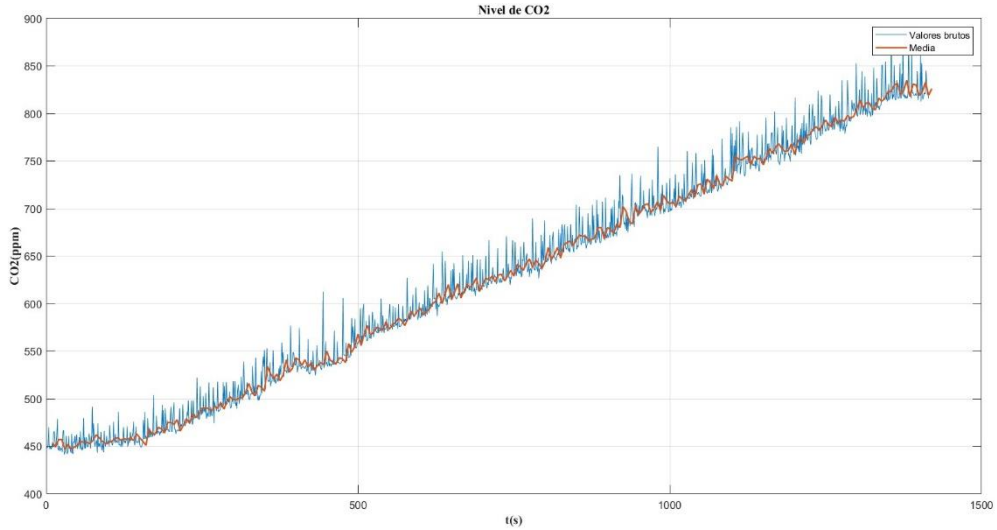


Figura 54. Nivel de CO2 a lo largo del tiempo (caso 4).

Si observamos la línea roja, podemos ver que el máximo se produce a los 1400 segundos y tiene un valor de 835 ppm. El nivel de CO_2 dentro de la habitación tiene una tendencia ascendente debido a que no hay renovación de aire. Además, se puede observar que la pendiente es mayor que en el tercer caso, es decir, llegamos a valores superiores a los 800 ppm en menor tiempo y eso es debido a que hay una persona haciendo ejercicio dentro de la habitación.

5.1.5 Quinto caso de estudio.

En este quinto caso, se ha realizado una sesión aproximadamente de media hora en el que una persona realiza ejercicio mientras la puerta está abierta 83º y la ventana está completamente cerrada.

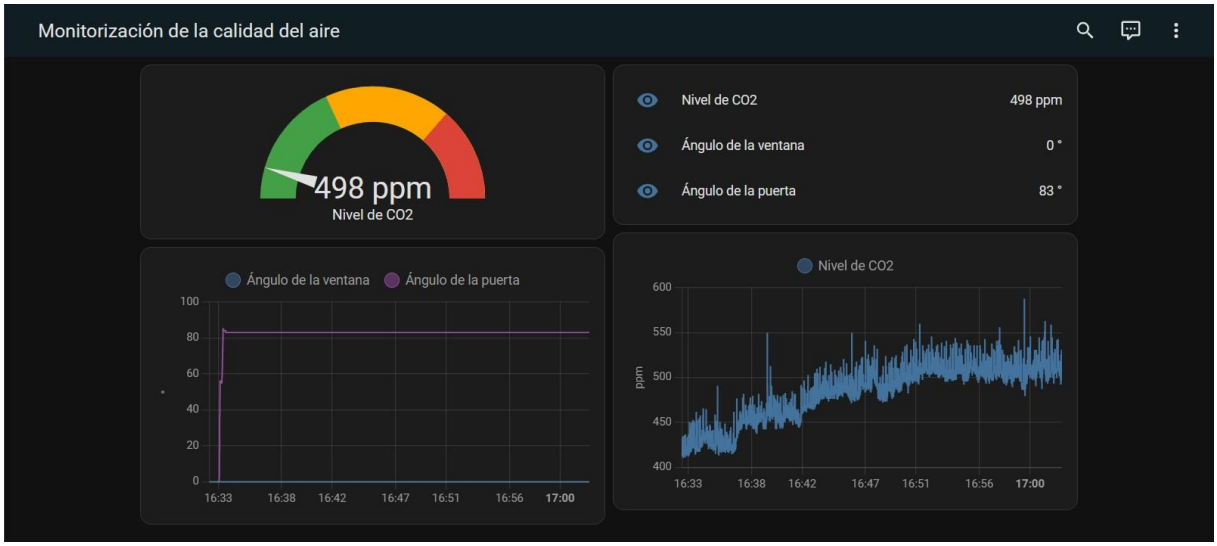


Figura 55. Dashboard en el quinto caso.

En la imagen se puede observar que la sesión empezó a las 16:33 y terminó a las 15:03. El nivel de CO_2 empieza oscilando en valores de 410-440 ppm y va aumentando hasta que se estabiliza oscilando en valores de 480-530 ppm con algún pico mayor a los 550 ppm. En este caso hay renovación de aire debido a que la puerta está abierta a 83º, por lo tanto, llega a un punto en el que el nivel de CO_2 se estabiliza. En la siguiente gráfica se puede apreciar mejor la tendencia del nivel de CO_2 que se representa, además de los valores brutos, la media de cada 5 muestras.

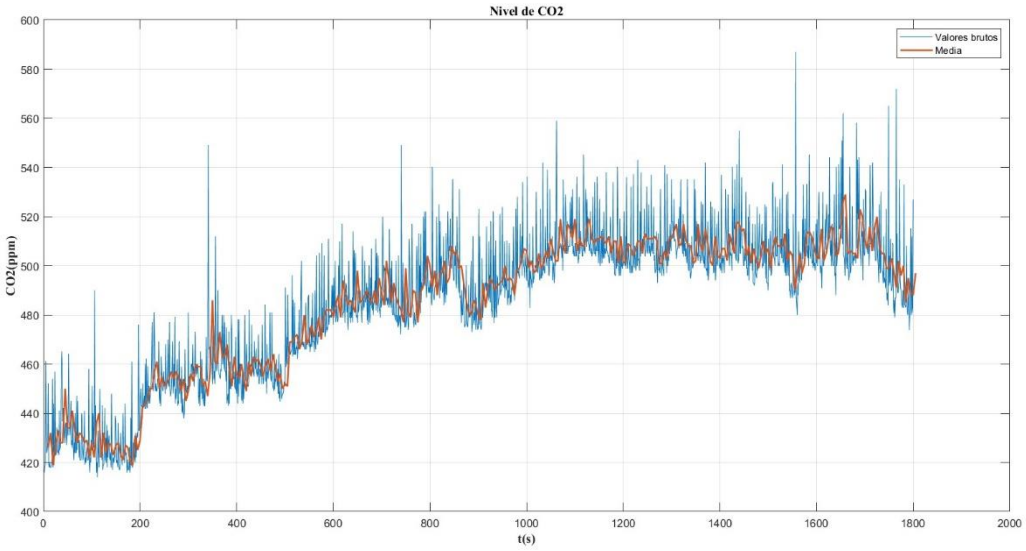


Figura 56. Nivel de CO2 a lo largo del tiempo (5º caso).

En la figura 54 se puede observar que se produce un máximo aproximadamente a los 1700 segundos y tiene un valor de 529 ppm. A pesar de la variabilidad producida por la alta frecuencia en la recolección de muestras se puede apreciar que llega un momento en el que el nivel de CO_2 se estabiliza en un valor aproximado a 500 ppm.

5.1.6 Sexto caso de estudio.

En el sexto caso de estudio se ha realizado una sesión aproximadamente de media hora en el que una persona realiza ejercicio mientras tanto la ventana como la puerta están abiertas 83º.

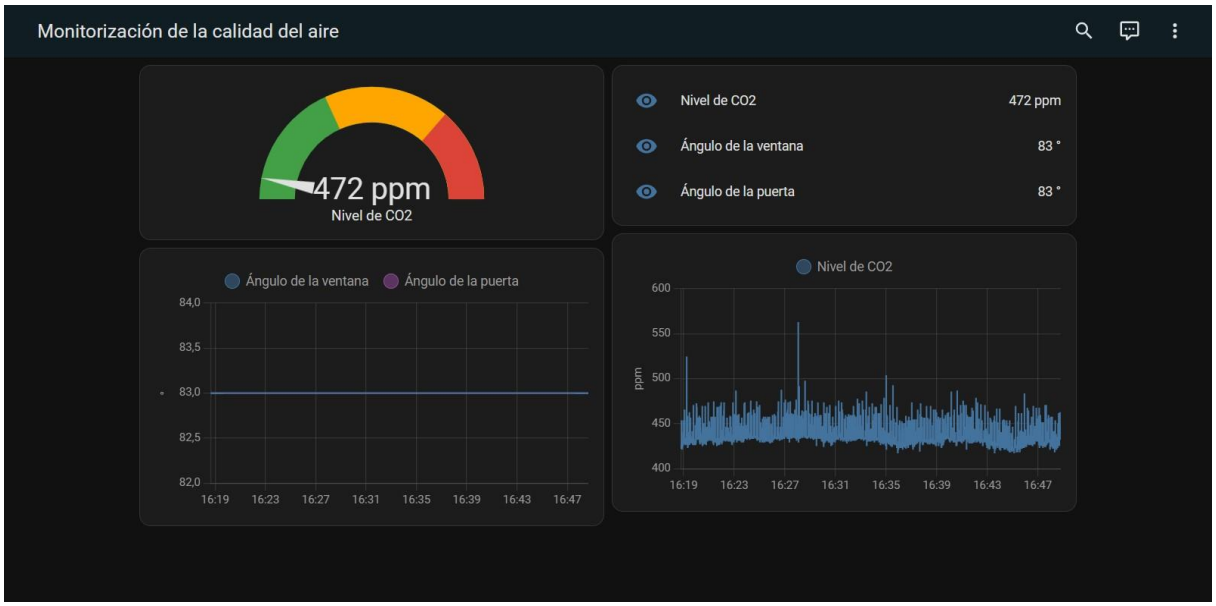


Figura 57. Dashboard en el sexto caso.

La sesión empieza a las 16:19 y termina a las 16:49. Durante esa media hora el nivel de CO2 oscila en valores de 410-450 ppm con algún pico que supera los 550 ppm. Aquí el nivel de CO_2 permanece estable oscilando entre los mismos valores y esto es debido a que hay una renovación de aire continua por lo que el CO_2 expulsado por la persona que está haciendo ejercicio apenas tiene efecto en este caso.

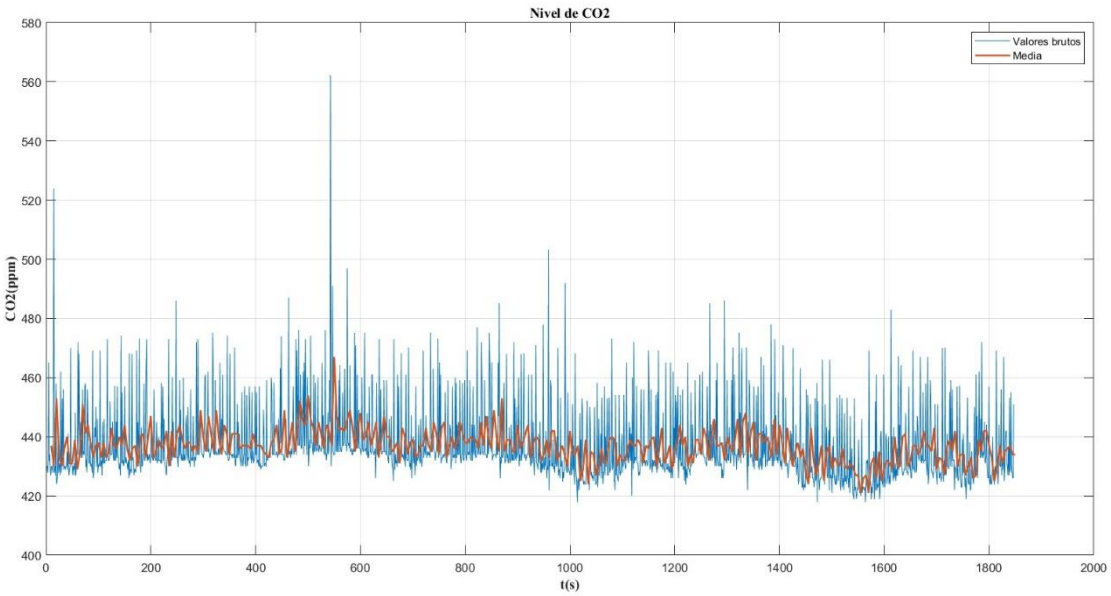


Figura 58. Nivel de CO2 a lo largo del tiempo (6º caso).

Si observamos la línea roja podemos apreciar que hay un máximo aproximadamente a los 580 segundos y tiene un valor de 467 ppm. Además, se puede ver que a lo largo de la media hora el nivel de CO_2 se ha estabilizado en valor aproximado de 435 ppm.

5.2 Evaluación de los resultados.

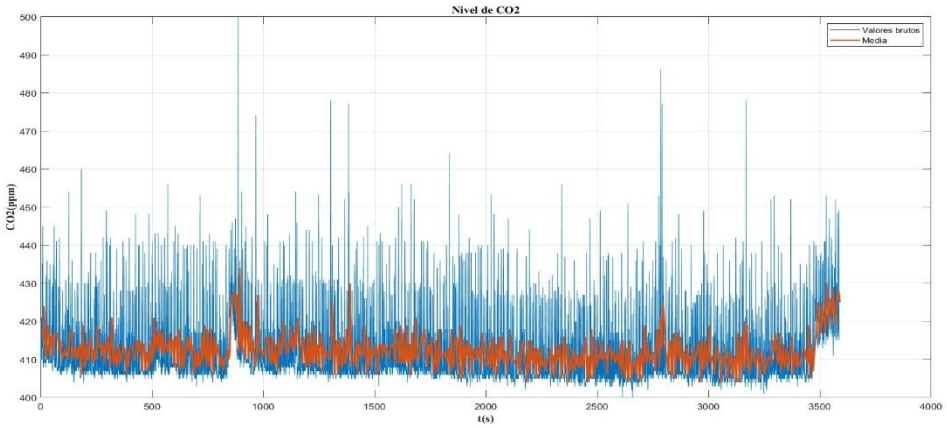
Lo primero que llama la atención en los diferentes casos estudiados es la variabilidad, que se puede apreciar en las diferentes gráficas, en el nivel de CO_2. Hay diferencia entre un resultado y el siguiente. Estas diferencias pueden ser debido a:

- Utilización del Wi-Fi: Cuando la placa se conecta al Wi-Fi, con el objetivo de enviar al nodo sumidero los datos obtenidos, utiliza los pines ADCs para medir voltajes internos y de esta manera controlar la potencia de salida. La utilización de estos pines hace que la función analogRead(), utilizada en el software diseñado para la obtención del nivel de CO_2, no mida correctamente el voltaje.
- El posicionamiento de algún cable puede afectar las mediciones.

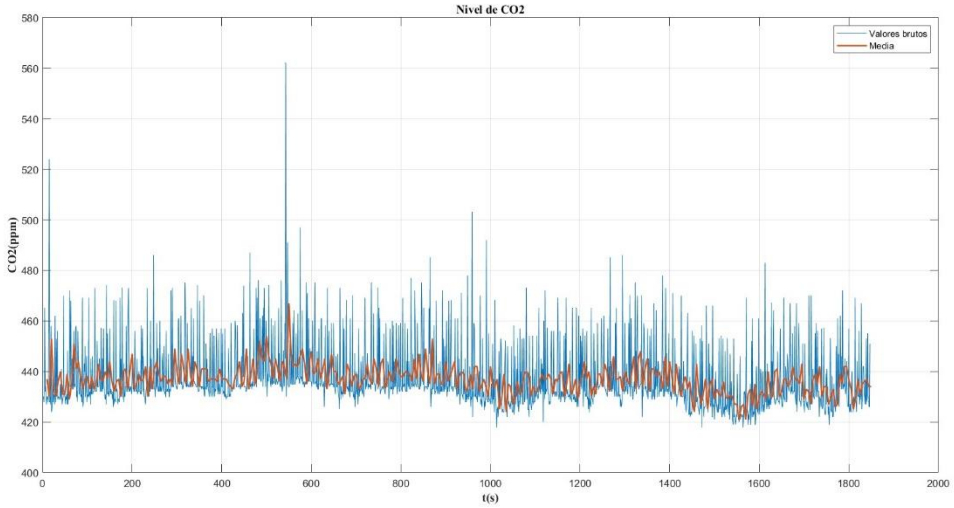
- Las características ambientales también pueden afectar las mediciones obtenidas por el sensor de gas SEN0159.

A pesar de esta variabilidad, el trazado que sigue cada gráfica es diferente para cada caso. A continuación, se va a comparar los resultados obtenidos en los diferentes casos:

- Ventana y puerta abierta 83º.



a) Nivel de CO2 con persona acostada.



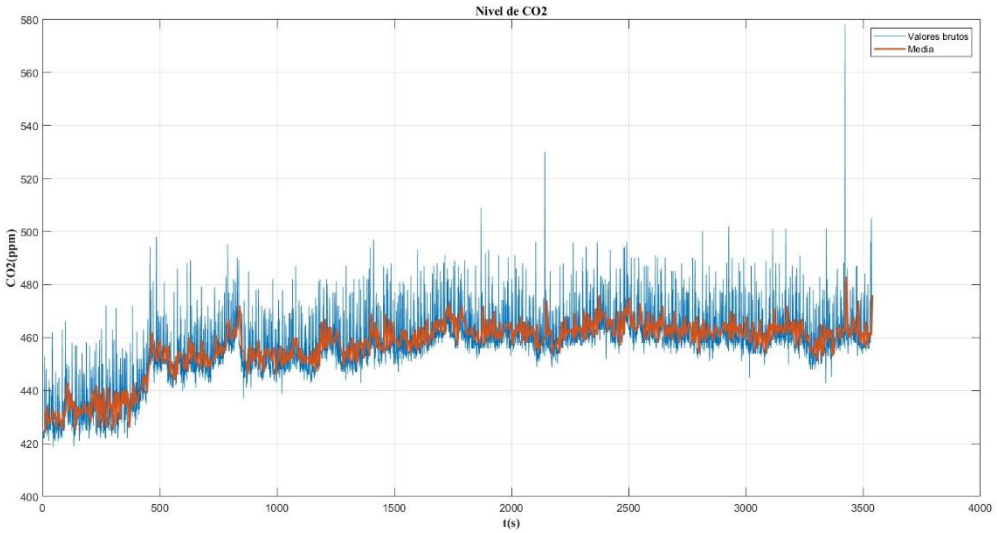
b) Nivel de CO2 con persona corriendo.

Figura 59. Comparativa del nivel de CO2 cuando la ventana y la puerta están abiertas 83º.

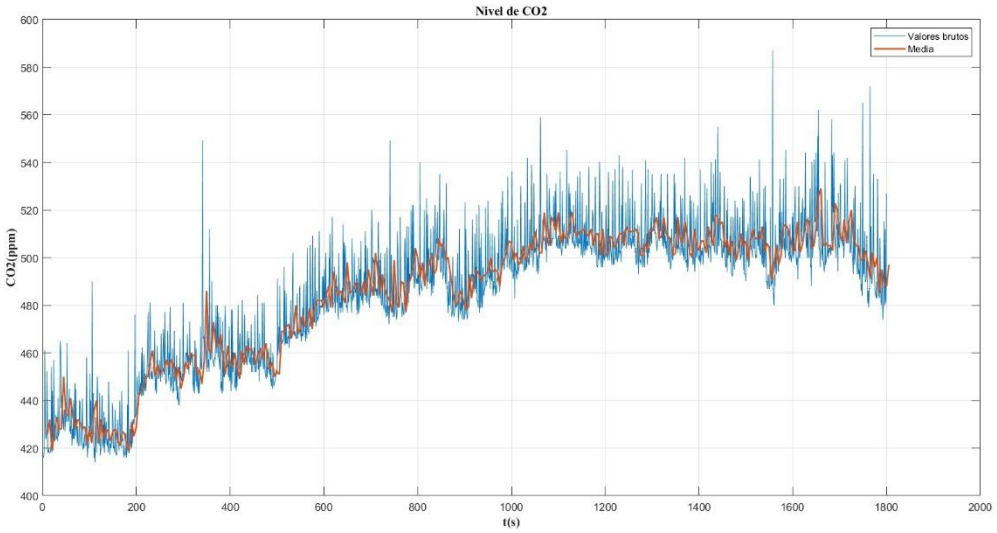
Cuando la ventana y la puerta del cuarto están abiertas 83º se puede observar, en los dos casos, que las gráficas siguen una trayectoria constante, es decir, el nivel de CO_2 permanece constante. Sin embargo, el nivel de CO_2 es distinto en ambos casos. Cuando hay una persona acostada, el nivel de CO_2 permanece constante, dentro de la variabilidad, en un rango de valores de 408-420 ppm con algún pico de más de 430 ppm. Sin embargo, cuando hay una persona corriendo el nivel de CO_2

permanece constante en un rango de valores de 430-450 con algún pico de más de 460 ppm. Como ya he comentado anteriormente, esta trayectoria constante es debido a que al estar la habitación completamente abierta la renovación de CO_2 en la habitación es constante. A pesar de esto, el prototipo diseñado para evaluar la calidad del aire en interiores detecta que el nivel de CO_2 es ligeramente superior cuando hay una persona corriendo en la habitación.

- Ventana abierta 83º y puerta cerrada.



a) Nivel de CO2 con persona acostada.

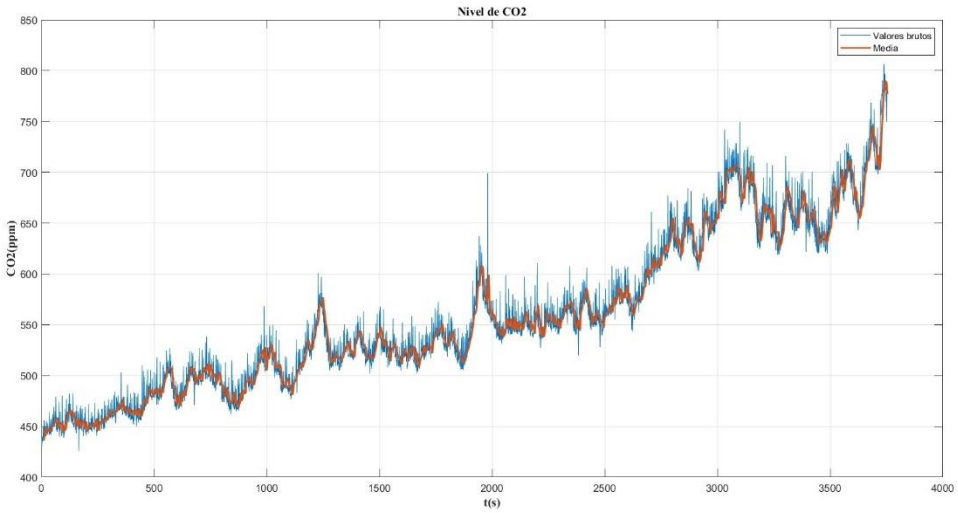


b) Nivel de CO2 con persona corriendo.

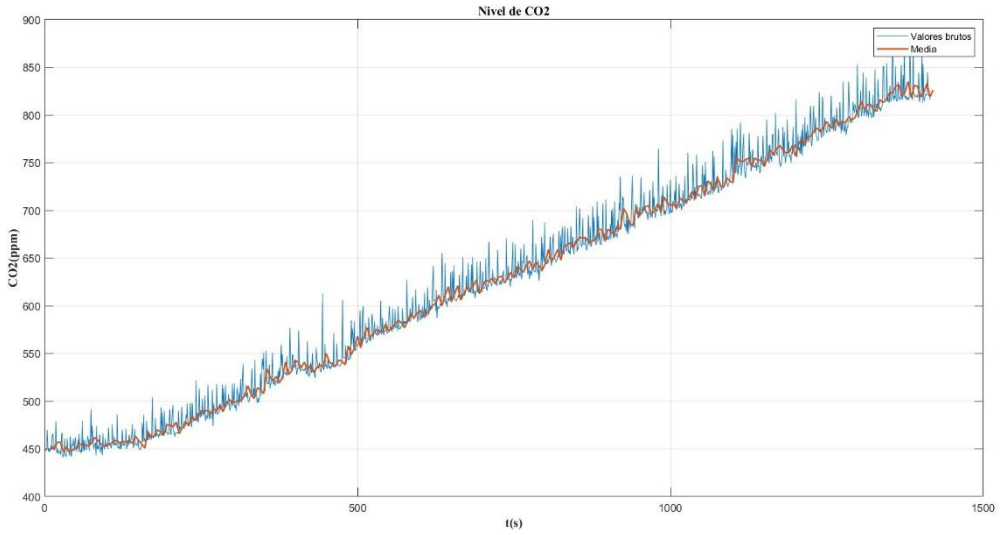
Figura 60. Comparación del nivel de CO2 cuando la ventana esta abierta 83º y la puerta cerrada.

Cuando la ventana está abierta 83º y la puerta está cerrada se puede observar que, en ambos casos, el nivel de CO_2 aumenta hasta que llega un momento donde se estabiliza oscilando entre ciertos valores. Cuando hay una persona acostada, el nivel de CO_2 se estabiliza oscilando en valores de 450-470 ppm aproximadamente. Cuando hay una persona corriendo, el nivel de CO_2 se estabiliza oscilando en valores de 490-510 ppm. A raíz de estos resultados, podemos decir que el prototipo diseñado, de nuevo, detecta que el nivel de CO_2 ha subido ligeramente cuando hay una persona corriendo dentro de la habitación.

- Ventana y puerta cerrada.



b) Nivel de CO2 con persona acostada.



a) Nivel de CO2 con persona corriendo.

Figura 61. Comparación del nivel de CO2 cuando la puerta y la ventana están cerradas.

Cuando la ventana y la puerta están cerradas, el nivel de CO_2, en ambos casos, aumenta sin parar debido a que no hay renovación de aire. Sin embargo, la velocidad de aumento del nivel de CO_2 es muy superior en el segundo caso debido a que hay una persona corriendo expulsando constantemente CO_2. Cuando la persona esta acostada, en una hora se llega a los 800 ppm, mientras tanto, cuando la persona esta corriendo en 25 minutos, aproximadamente, el nivel de CO_2 ya ha llegado a los 800 ppm.

6. Conclusión.

En este proyecto se ha presentado las diferentes fases que se han llevado a cabo para el desarrollo de un sistema IoT de monitorización de la calidad del aire en interiores. Para ello, primero hemos pasado por una fase de planificación. En esta fase se ha pensado los objetivos a cumplir, se ha organizado como se iba a desarrollar el proyecto y se ha realizado un presupuesto. Tras esta primera fase de planificación, se ha realizado el diseño el diseño de la arquitectura IoT. Para ello, se ha tenido que hacer una selección del hardware a utilizar, programar las diferentes placas de desarrollo utilizadas, diseñar los soportes para estas placas, implementar una base de datos en MongoDB para la persistencia de los datos en el tiempo y la utilización de una plataforma IoT, en nuestro caso Home Assistant, para la visualización de los datos en tiempo real.

Se puede decir que el trabajo ha sido exitoso ya que se han cumplido los objetivos marcados al inicio de este. El prototipo diseñado muestra en tiempo real tanto el nivel de CO_2 como el ángulo de apertura de la ventana y de la puerta. Los resultados obtenidos muestran como el nivel de CO_2 varía según cambia el ángulo de la puerta y/o ventana.

Cabe mencionar los problemas que nos podemos encontrar trabajando con este tipo de sensores ya que miden características analógicas del mundo y, a veces, la posición de un cable o las características ambientales pueden afectar los resultados obtenidos causando anomalías.

A pesar de cumplir los requisitos iniciales, al prototipo diseñado se le puede realizar mejoras. Algunas de las mejoras que puede ser útiles para ampliar en un futuro esta línea de trabajo son:

- Utilización de sensores inerciales que, además de giroscopio y acelerómetro, tenga magnetómetro para la identificación del norte. Por ejemplo, se podría

utilizar el sensor MPU9250 en vez del sensor MPU6050. De esta forma, nos ahorraríamos el diseño de los soportes ya que se podría colocar el sensor inercial directamente en la ventana y/o puerta sin tener que estar totalmente horizontal.

- Incorporación de un sensor de temperatura y un sensor de humedad ya que son parámetros muy importantes para la evaluación de la calidad del aire.
- Añadir acciones de forma proactiva. Se puede automatizar el cierre y/o la apertura tanto de la ventana como de la puerta dependiendo del nivel de CO_2 que hay en la habitación.
- Incorporar un sistema de conteo de persona ya que la acumulación de personas en la habitación aumenta el nivel de CO_2.
- Análisis de los datos obtenidos por medio de modelos de ciencia de datos para obtener información más significante.

La realización de este TFM me ha dado la oportunidad de adentrarme y aprender de un mundo totalmente alejado de mi disciplina. He podido explorar en profundidad conceptos como las plataformas IoT, la base de datos o los protocolos de comunicación.

7. Apéndices.

En este apartado se describirá paso a paso como configurar la IDE de Arduino para su funcionamiento con la ESP8266, la instalación del sistema operativo Raspberry Pi OS y la configuración del programa PyCharm Community. Además, se mostrarán todos los códigos utilizados en este proyecto y los planos de los modelos 3D diseñados.

7.1 Manual de configuración de la IDE de Arduino para su funcionamiento con la ESP8266.

Primero, se debe instalar la IDE de Arduino en nuestro ordenador. Para ello, nos lo descargaremos de la página web oficial de Arduino (hacer click aquí). Una vez se ha instalado la IDE de Arduino en nuestro ordenador, pasamos a configurar esta con el objetivo de utilizarla para nuestra placa ESP8266. Para realizar el primer paso de la configuración de nuestra IDE de Arduino, tenemos que ir a la ventana de Preferencias. A esta ventana accedemos pulsando Archivo→Preferencias del menú situado en la parte superior de nuestra IDE. En esta ventana debemos pegar, en la sección 'Gestor de URLs Adicionales de Tarjetas', lo siguiente:

http://arduino.esp8266.com/stable/package_esp8266com_index.json

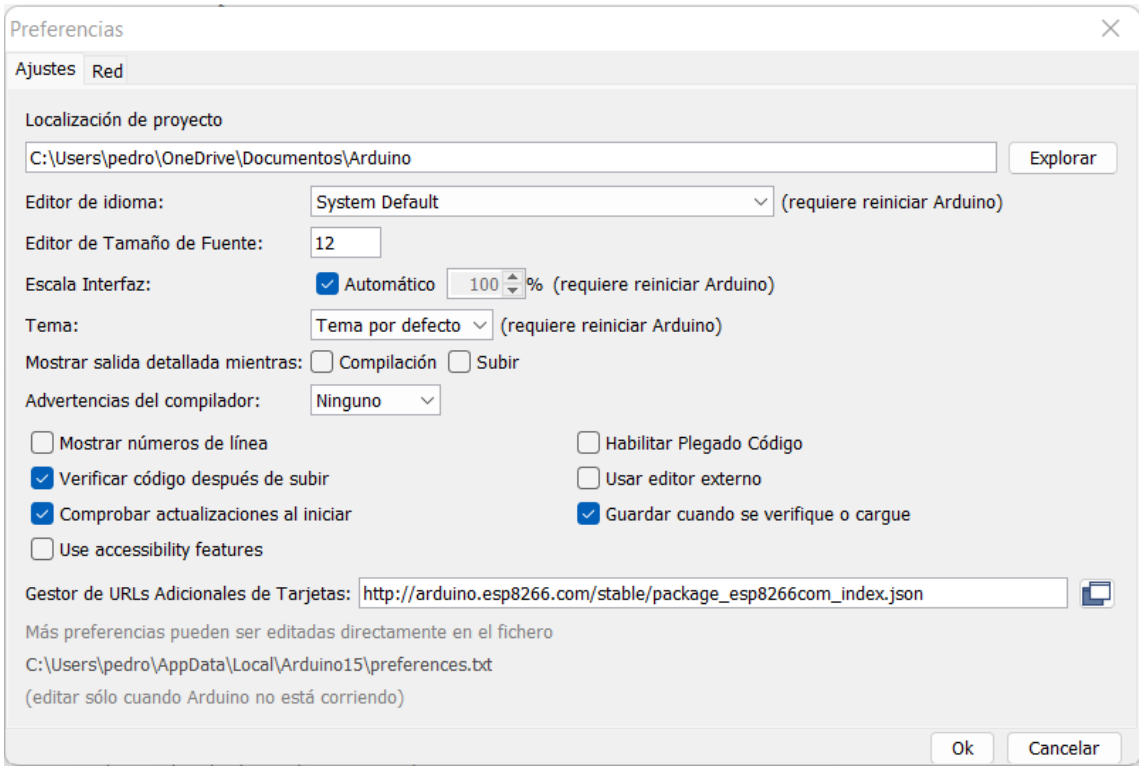


Figura 62. Ventana Preferencias en la IDE de Arduino (ESP8266).

Una vez realizado este primer paso, debemos incluir la placa ESP8266 en el Gestor de tarjetas. Para acceder al Gestor de tarjetas debemos pulsar Herramientas→Placa→Gestor de tarjetas, como se muestra en la siguiente figura:

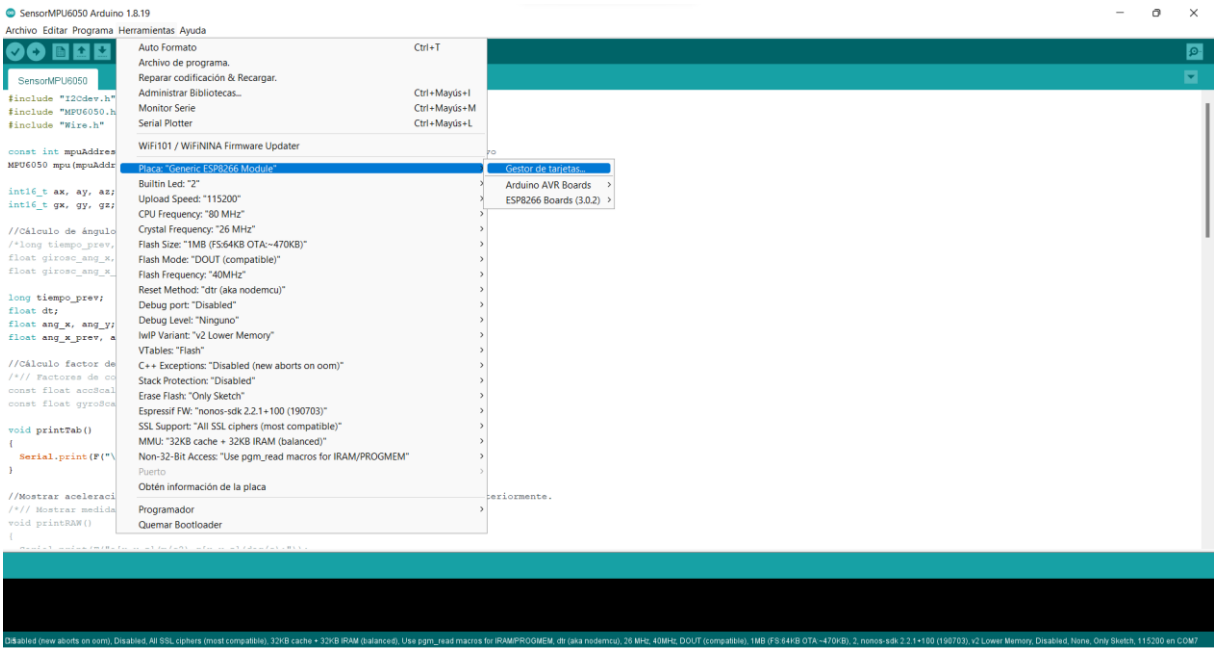


Figura 63. Acceso al Gestor de tarjetas en la IDE de Arduino.

En la ventana Gestor de tarjetas debemos buscar 'ESP8266' e instalarla, pulsando el botón instalar.

Figura 64. Búsqueda de ESP8266 en el Gestor de tarjetas.

Por último, se debe seleccionar, como placa, la opción ‘Generic ESP8266 Module’, pulsando Herramientas→Placa→ESP8266 boards→Generic ESP8266 Module.

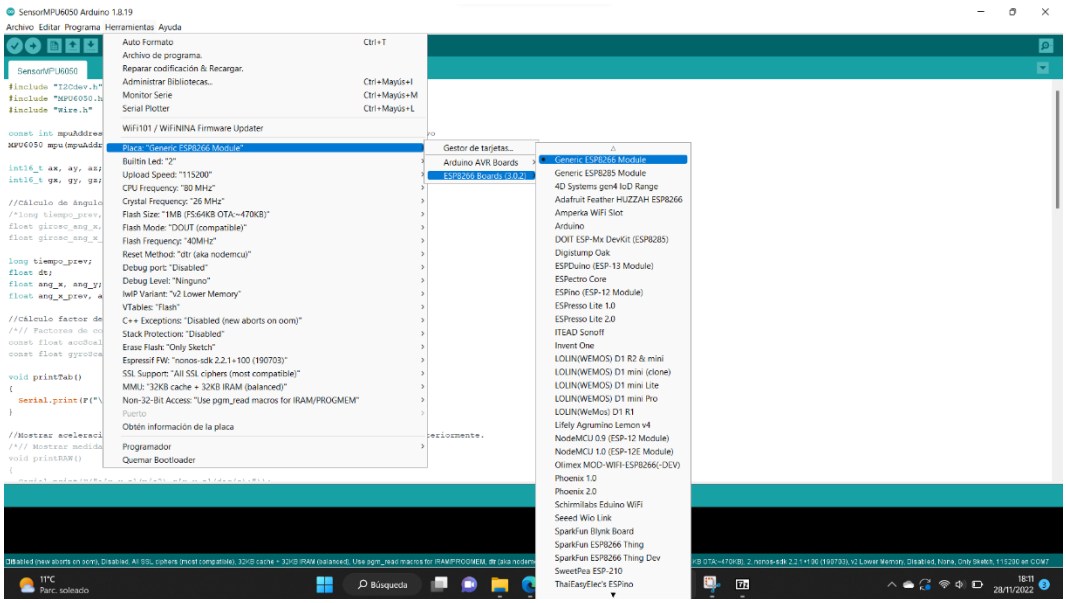


Figura 65. Elección de la placa 'Generic ESP8266 Module'.

7.2 Manual de configuración de la IDE de Arduino para su funcionamiento con la ESP32.

Primero, se debe instalar la IDE de Arduino en nuestro ordenador. Para ello, nos lo descargaremos de la página web oficial de Arduino (hacer click aquí). Una vez se ha instalado la IDE de Arduino en nuestro ordenador, pasamos a configurar esta con el objetivo de utilizarla para nuestra placa ESP32. Para realizar el primer paso de la configuración de nuestra IDE de Arduino, tenemos que ir a la ventana de Preferencias. A esta ventana accedemos pulsando Archivo→Preferencias del menú situado en la parte superior de nuestra IDE. En esta ventana debemos pegar, en la sección ‘Gestor de URLs Adicionales de Tarjetas’, lo siguiente:

https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json

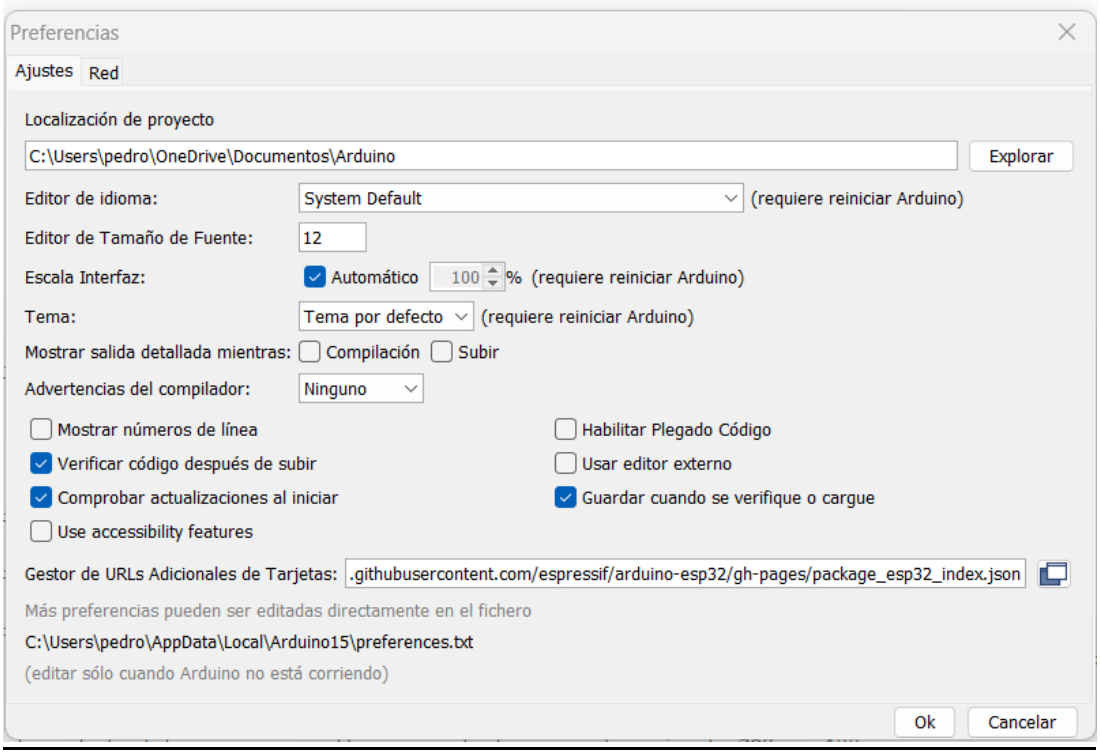


Figura 66. Ventana Preferencias en la IDE de Arduino (ESP32).

Una vez realizado este primer paso, debemos incluir la placa ESP8266 en el Gestor de tarjetas. Para acceder al Gestor de tarjetas debemos pulsar Herramientas→Placa→Gestor de tarjetas, como se muestra en la figura 56. En la ventana Gestor de tarjetas debemos buscar 'ESP32' e instalarla, pulsando el botón instalar.

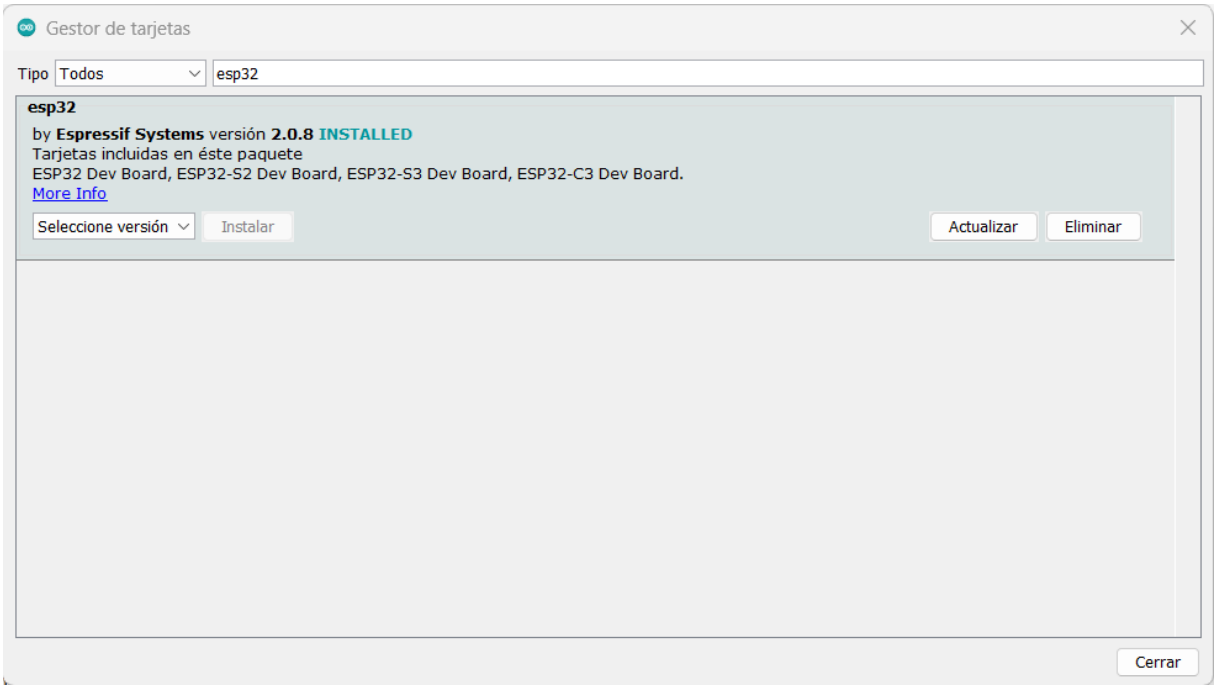


Figura 67. Búsqueda de ESP32 en el gestor de tarjetas.

Por último, se debe seleccionar, como placa, la opción 'ESP32 Dev Module', pulsando Herramientas→Placa→ESP32 Arduino→ESP32 Dev Module

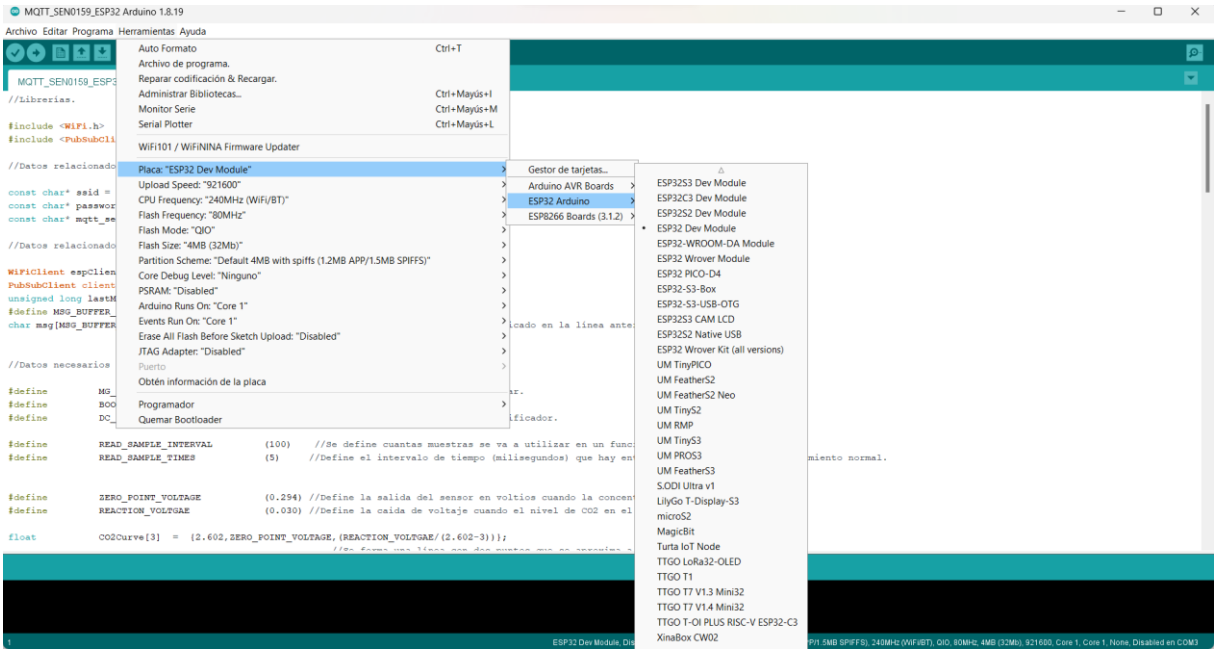


Figura 68. Elección de la placa 'ESP32 Dev Module'

7.3 Instalación del sistema operativo en la Raspberry Pi 3.

Se necesita la tarjeta microSD, que introduciremos posteriormente en la Raspberry, un lector de tarjetas y un ordenador.

Para la instalación del sistema operativo Raspberry Pi OS, primero, debemos descargar el software Raspberry Pi Imager desde la página web de Raspberry (se puede descargar aquí) e introducir la tarjeta microSD en el lector de tarjetas. A continuación, se debe ejecutar Raspberry Pi Imager.

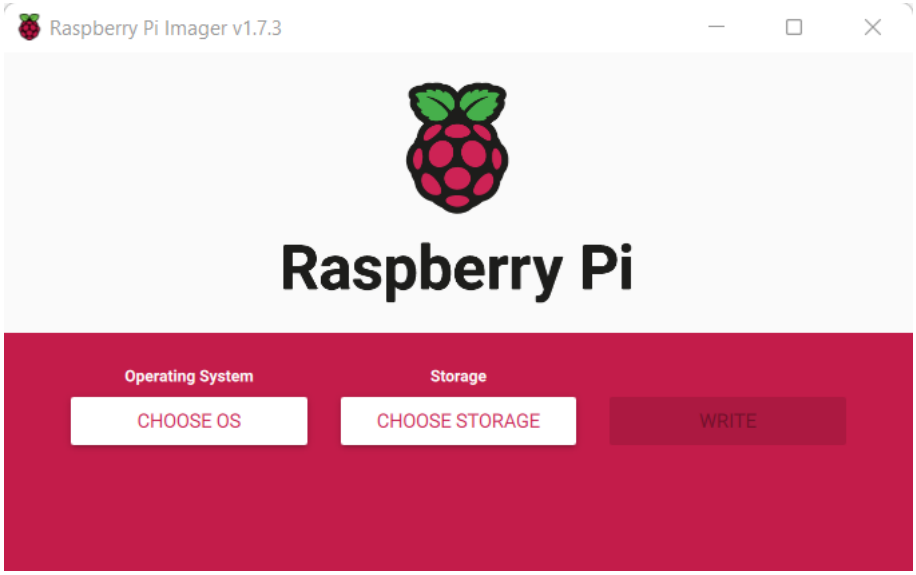


Figura 69. Captura Raspberry Pi Imager.

Ahora se debe elegir el sistema operativo pulsando CHOOSE OS. En nuestro caso, hemos elegido el sistema operativo Raspberry Pi OS lite (64 bit) (“A port of Debian Bullseye with no desktop environment”). Este sistema operativo se encuentra haciendo click en la opción Raspberry Pi OS (other).

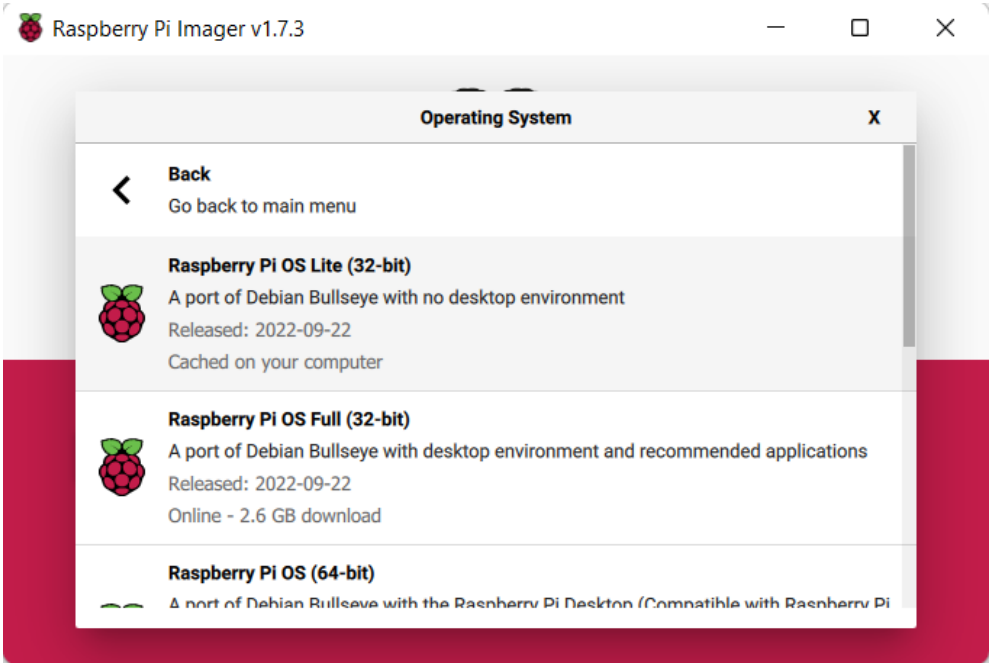


Figura 70. Elección del sistema operativo.

Tras elegir el sistema operativo, se debe elegir la unidad de almacenamiento, en nuestro caso, la tarjeta microSD que hemos introducido en el lector de tarjetas. Para ello, debemos pulsar CHOOSE STORAGE. A continuación, se debe pulsar WRITE para terminar la instalación del sistema operativo.

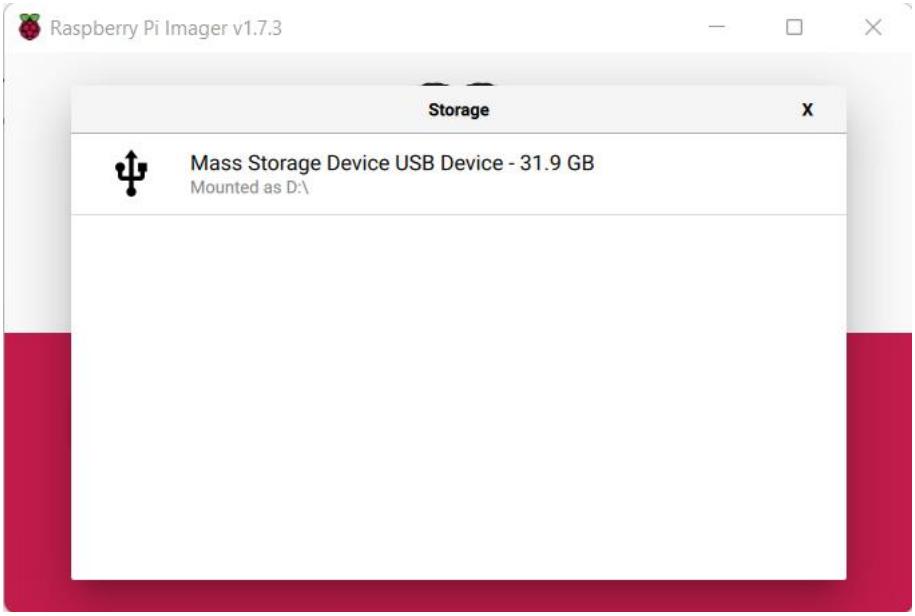


Figura 71. Elección de la unidad de almacenamiento.

Una vez instalado el sistema operativo, necesitamos activar el servicio SSH. El servicio SSH es un protocolo de comunicación que nos permite el acceso remoto a la Raspberry y así evitar tener que conectar un teclado y un monitor cada vez que debamos realizar cambios en las configuraciones del servidor.

Para activar el servicio SSH sólo debemos crear un archivo llamado SSH en nuestra tarjeta microSD sin extensión. Sin embargo, es necesario estar conectado a la red con la Raspberry Pi para que se active este servicio. En este caso, aunque disponemos de conexión vía Ethernet, se ha decidido utilizar un punto de conexión wifi creado desde el móvil para conectar nuestra Raspberry a la red. Una vez creado el punto de acceso en nuestro móvil, se debe crear un archivo llamado wpa_supplicant.conf en el directorio boot, es decir, en nuestra tarjeta microSD. Este archivo debe tener en su interior el siguiente contenido:

```
country=CA
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1

network= {
    ssid="wifi name"
    psk="wifi password"
}
```

En CA debemos poner, en nuestro caso, ES, ya que se ha trabajado en España. Además, debemos introducir el nombre del wifi que hemos creado en nuestro móvil en “wifi name” y la contraseña en “wifi password”.

Una vez finalizado toda esta configuración, si queremos acceder a la Raspberry a través de nuestro ordenador, primero debemos introducir la tarjeta microSD en la ranura de tarjetas que hay en la Raspberry Pi. Conectamos la Raspberry a nuestro ordenador y observaremos un led encendido que nos indica que esta funcionando. Ahora sólo debemos abrir el CMD en Windows y poner la siguiente línea:

```
ssh pi@ip
```

En ip debemos colocar la IP de nuestra Raspberry Pi. En este caso, se ha podido mirar esta IP a través del móvil. Pero si nuestro celular no nos permite mirar la IP de los dispositivos que se están conectando al punto de acceso que hemos creado, se puede utilizar el programa wnetwatcher.ip en nuestro ordenador. Este programa nos da las IPs de los dispositivos que están conectados a la red.

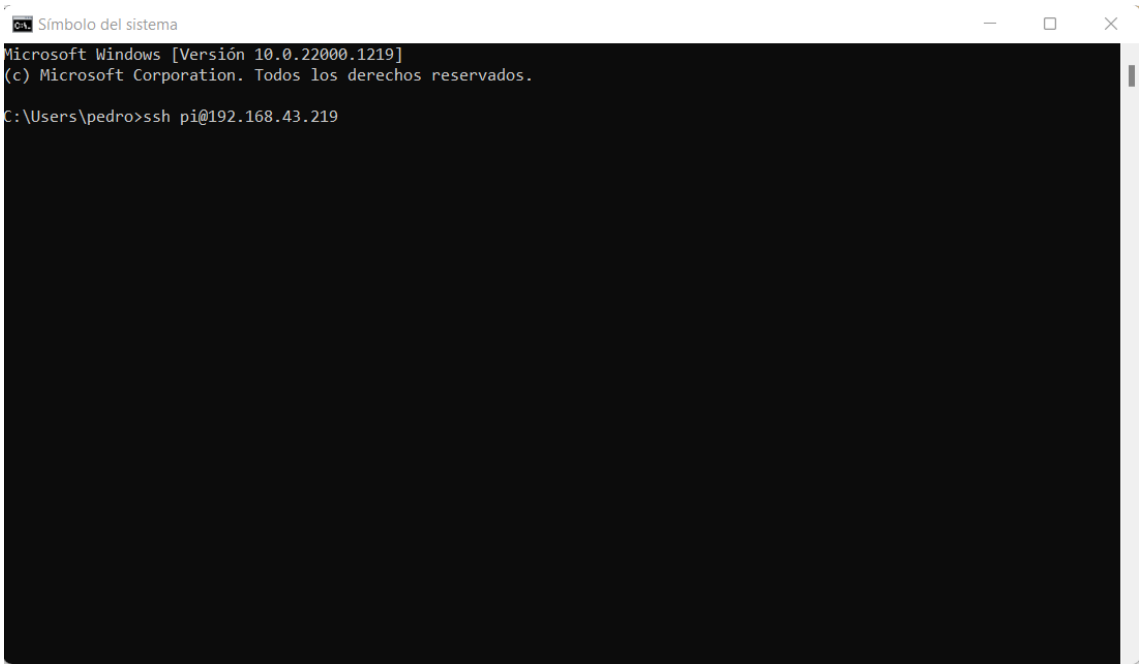


Figura 72. Línea de código para conectarse a través de SSH.

Cuando introducimos el comando anterior, se nos pide una contraseña. Esta contraseña, por defecto, es raspberrypi.

Para poder acceder a la Raspberry Pi a través de nuestro ordenador es importante estar conectado en el ordenador a la misma red Wifi a la que hemos conectado la Raspberry Pi.

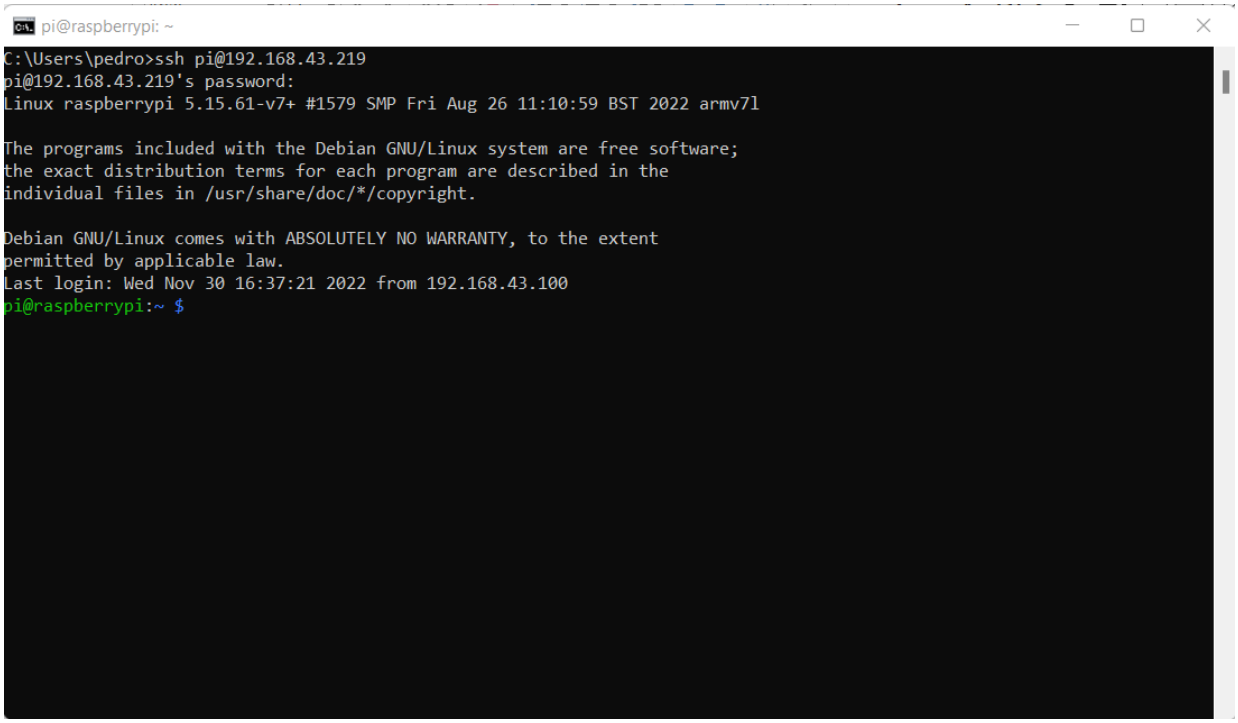


Figura 73. Conexión por SSH a la Raspberry Pi.

7.4 Instalación de librerías en PyCharm.

Para recoger los datos en la base de datos MongoDB se ha realizado un programa en PyCharm Community en lenguaje Python. En este programa se ha utilizado 4 librerías:

- Paho-mqtt: permite crear un cliente MQTT, con el que podrás suscribirte o publicar un topic.
- Pytz: Habilita el cálculo de forma precisa de las zonas horarias del mundo en Python.
- Datetime: Permite trabajar en Python con fechas y horas.
- Pymongo: Contiene herramientas que permiten la interacción con MongoDB mediante Python.

Una vez tenemos instalado PyCharm Community en nuestro ordenador, se procede a instalar estas librerías. Para ello, debemos pulsar File→Settings. En la ventana Settings debemos hacer click en Python Interpreter como se muestra en la siguiente figura:

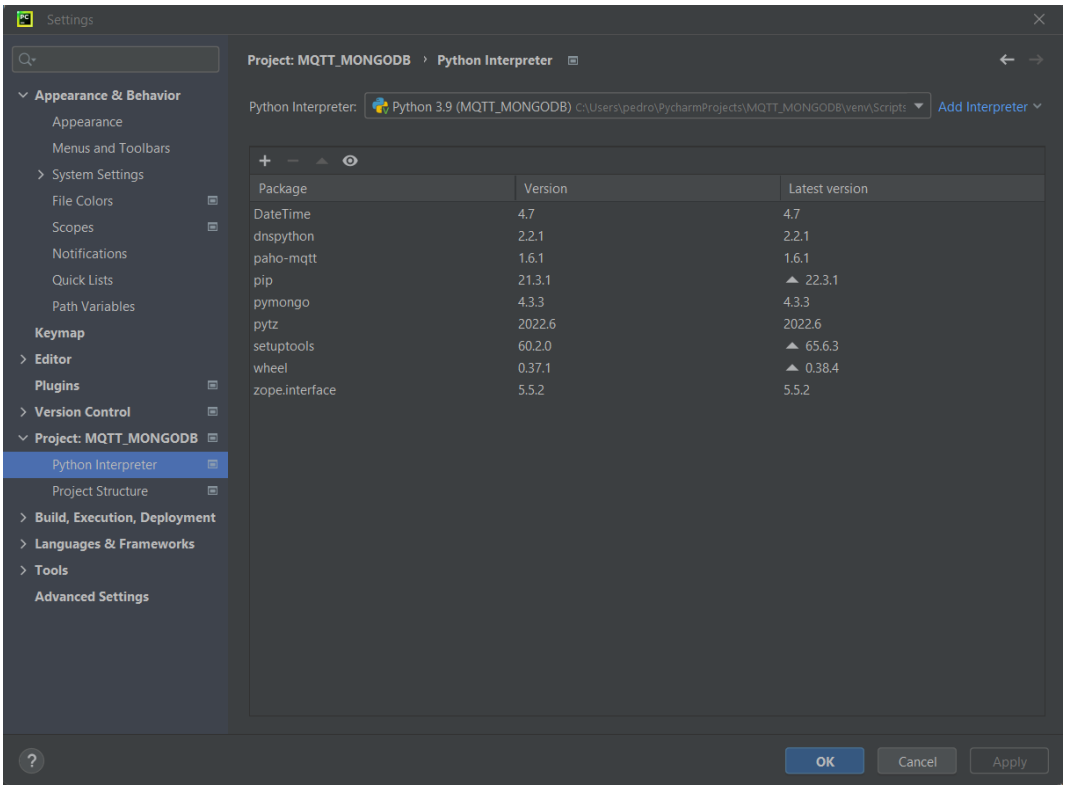


Figura 74. Ventana Settings en PyCharm Community.

Dentro de Python Interpreter debemos pulsar el signo más para agregar una librería. En la ventana que se abre, buscamos las diferentes librerías que necesitamos y las instalamos pulsando Install Package.

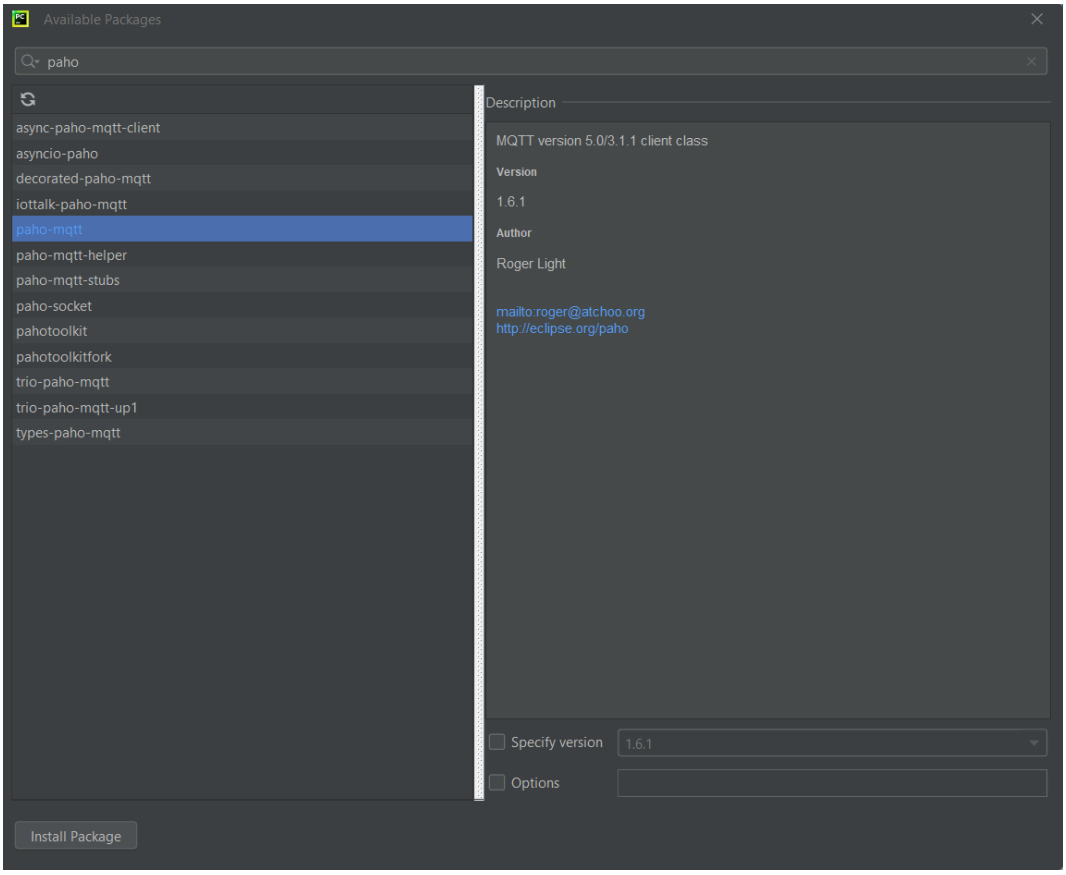


Figura 75. Instalación de la librería paho-mqtt.

7.5 Instalación de Home Assistant Supervised en Raspberry Pi 3.

Para la realización de esta instalación es muy importante acceder a la Raspberry Pi a través del cable ethernet. Si se realiza la instalación mientras estamos conectados a la Raspberry Pi mediante wifi, se producirá un error en mitad de la instalación debido a que, durante esta, hay un reinicio del NetworkManager que desconecta el wifi en la Raspberry Pi lo que provoca una desconexión de esta y por lo tanto que no se termine de instalar el software.

Para acceder a la Raspberry Pi mediante cable Ethernet se necesitará:

- Un cable Ethernet que conecte la Raspberry Pi con un router.
- Un cable Ethernet que conecte el ordenador, donde estamos trabajando, con el mismo router.
- Una Raspberry Pi.

- Un ordenador al que se le pueda conectar un cable Ethernet. Si el ordenador no se le puede conectar este cable deberemos conseguir un adaptador de red para poder conectarlo.
- Cargador de la Raspberry Pi.
- Haber activado el protocolo SSH, como se ha explicado anteriormente en el punto 6.2.

Para acceder a la Raspberry Pi, primero, se conectan los dos cables Ethernet tanto al ordenador como a la Raspberry Pi. Una vez esta conectados los cables Ethernet, se enchufa el cargador de la Raspberry Pi a esta para encenderla. Al enchufar nuestra Raspberry a la corriente se encenderán dos leds, uno de color naranja y otro de color verde, donde está conectado el cable Ethernet. Ahora, debemos instalar el programa MobaXterm (se puede descargar aquí). Una vez instalado, lo ejecutamos, pulsamos Session→SSH y escribimos raspberrypi.local en el campo Remote host como se muestra en la siguiente figura:

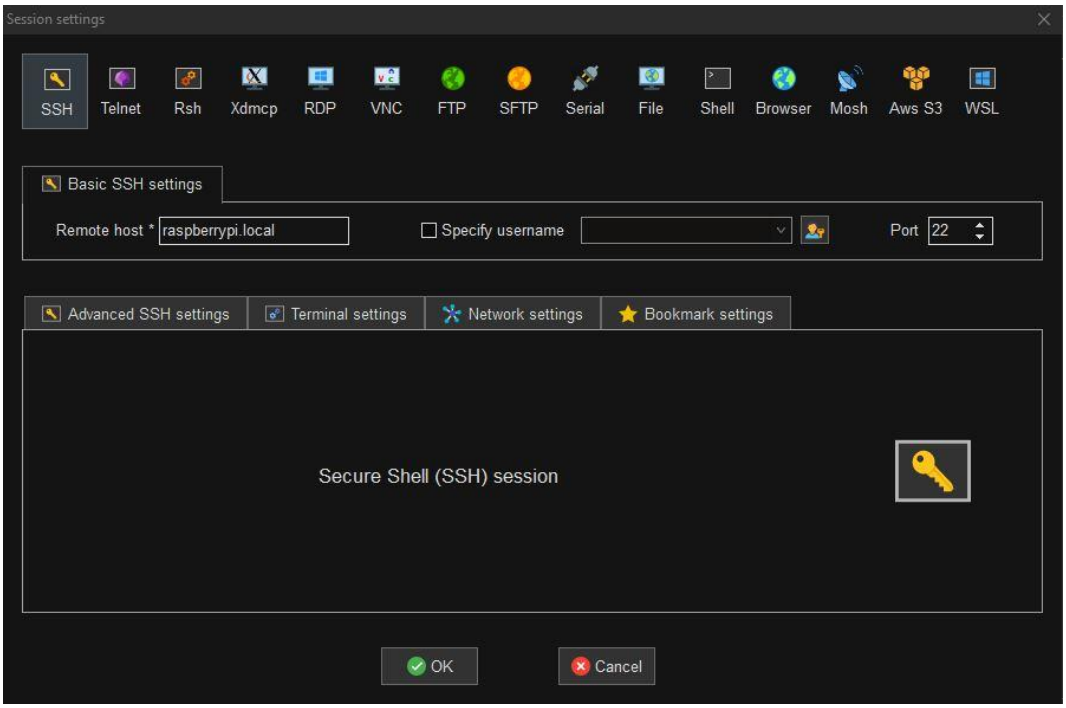


Figura 76. Acceso Raspberry Pi mediante el programa MobaXterm

Ahora solo debemos pulsar OK y el programa nos pedirá un usuario y una contraseña. El usuario es pi y la contraseña es, por defecto, raspberrypi. Una vez hemos accedido a la Raspberry Pi debemos seguir los siguientes pasos para la instalación de Home Assistant Supervised:

1º. Actualizaremos el sistema escribiendo las siguientes líneas de código:

```
Sudo apt update
Sudo apt upgrade
```

2º. Instalaremos Docker y añadiremos el usuario pi al grupo Docker mediante las siguientes líneas de código:

```
Curl -fsSL https://get.docker.com -o get-docker.sh
Sh get-docker.sh
Sudo usermod -aG docker pi
```

3º. Debemos instalar dependencias de Home Assistant Supervised mediante las siguientes líneas de código:

```
Sudo apt-get install \
jq \
wget \
curl \
udisks2 \
libglib2.0-bin \
network-manager \
apparmor \
apt-transport-https \
ca-certificates \
dbus -y
```

4º. Instalaremos el paquete OS Agent a través de las siguientes líneas de código:

```
Wget https://github.com/home-assistant/os-
agent/releases/download/1.4.1/os-agent_1.4.1_linux_armv7.deb
Sudo dpkg -i os-agent_1.4.1_linux_armv7.deb
```

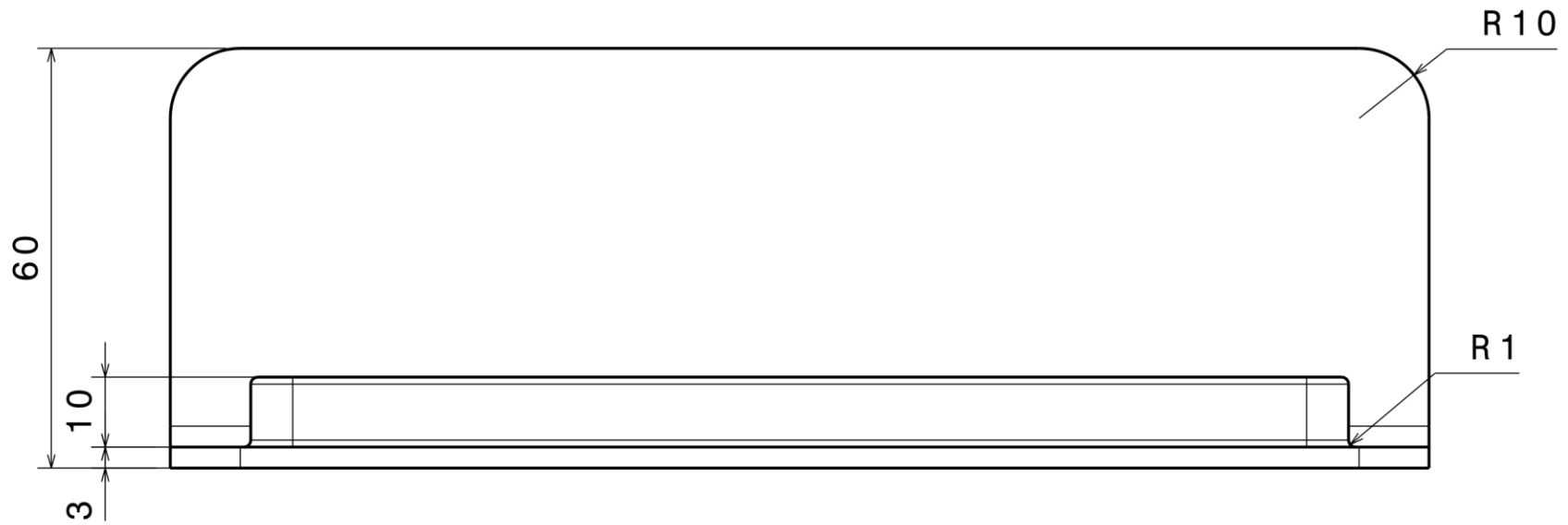
5º. Por último, se instala el paquete Home Assistant Supervised mediante las siguientes líneas de código:

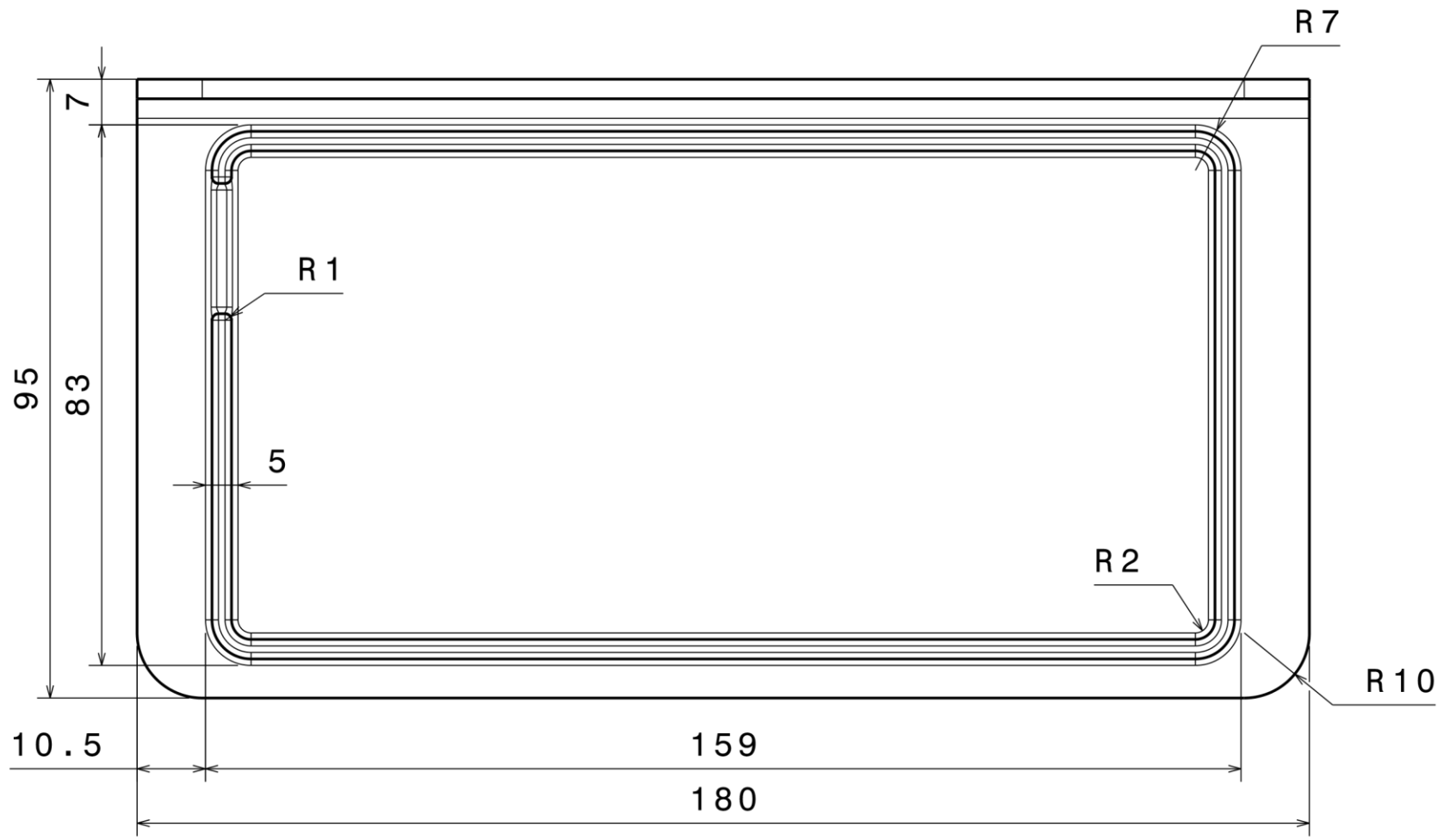
```
Wget https://github.com/home-assistant/supervised-
installer/releases/latest/download/homeassistant-supervised.deb
Sudo dpkg -i homeassistant-supervised.deb
```

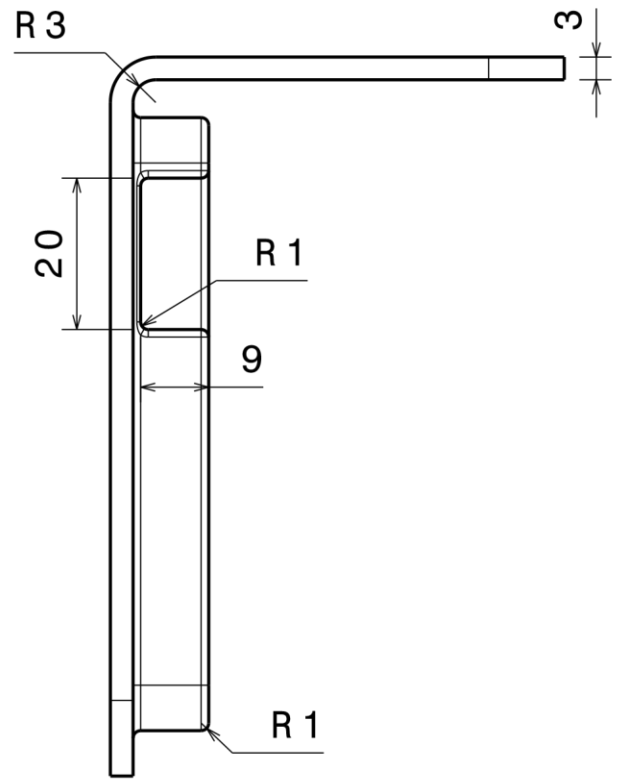
Una vez instalado, el programa nos dará una dirección para acceder Home Assistant. En mi caso, esta dirección es 192.168.1.142:8123. Si colocamos esa dirección en un navegador web accederemos a Home Assistant. El último paso para finalizar la instalación, es crearnos una cuenta en Home Assistant. Para ello, sólo debemos escribir un usuario y una contraseña en los campos pertinentes que aparecen la primera vez que accedemos.

7.6 Planos de los soportes diseñados.

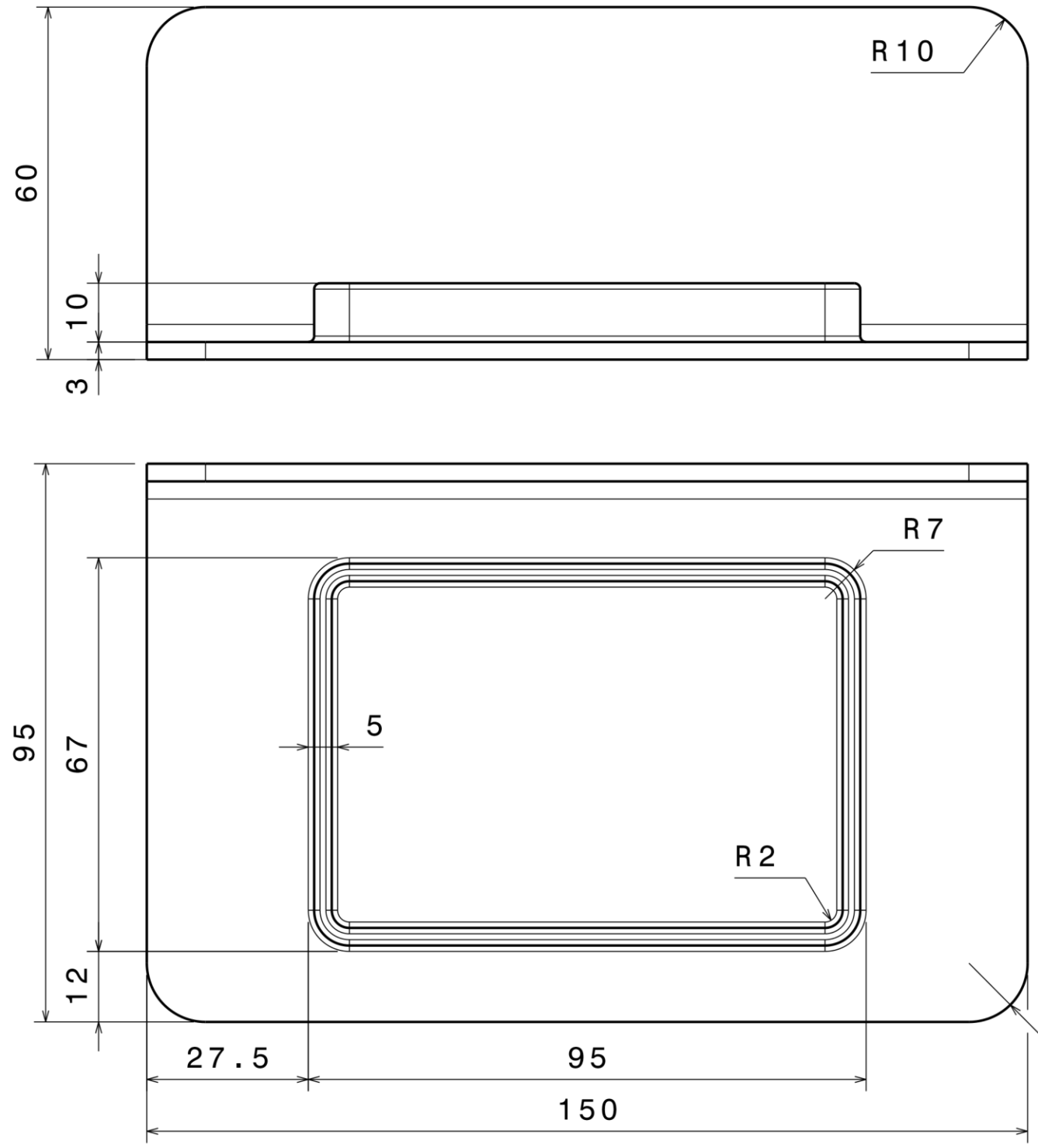
En este apartado se muestra las medidas para el diseño de ambos soportes:

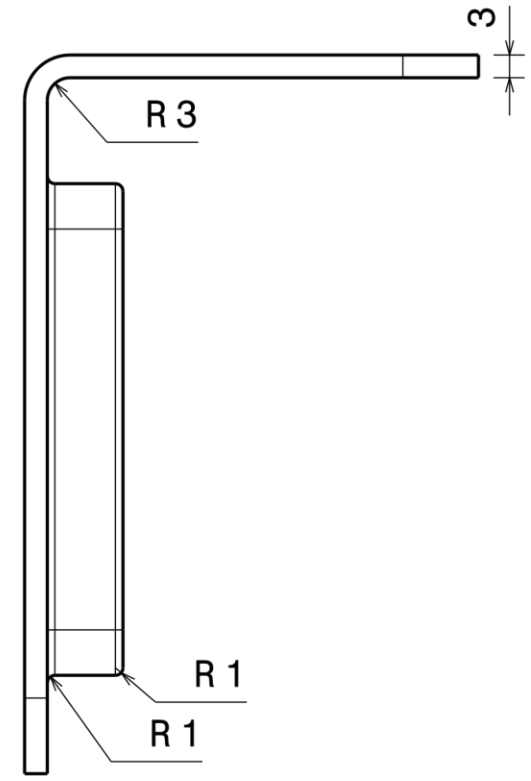






Universidad de Jaén		Universidad de Jaén		
DRAWN BY: Pedro José	DATE: 06/05/2023	DRAWING TITLE: Soporte		
CHECKED BY: Pedro José	DATE: 06/05/2023	SIZE: A3	DRAWING NUMBER	REV
DESIGNED BY: Pedro José	DATE: 20/04/2023	SCALE: 1:1	WEIGHT(kg): 0,09	SHEET: 1/1





Universidad de Jaén

DRAWN BY: Pedro José	DATE: 06/05/2023	Universidad de Jaén	
CHECKED BY: Pedro José	DATE: 06/05/2023	DRAWING TITLE: Soporte	
DESIGNED BY: Pedro José	DATE: 20/04/2023	SIZE: A3	DRAWING NUMBER:	REV:
SCALE: 1:1	WEIGHT(kg): 0.076		SHEET: 1/1

7.7 Proyectos alojados en las diferentes placas de desarrollo.

7.7.1 Proyecto alojado en la ESP8266.

En este TFM se han utilizado dos placas de desarrollo ESP8266, una para cada sensor de inercia. En la placa de desarrollo que había instalada en la puerta, el proyecto alojado era el siguiente:

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include "I2Cdev.h"
#include "Wire.h"
#include "MPU6050.h"


// Update these with values suitable for your network.

const char* ssid = "MIWIFI_2G_Hmmx";
const char* password = "EPpubggN";
const char* mqtt_server = "192.168.1.163";

// La dirección del MPU6050 puede ser 0x68 o 0x69, dependiendo
// del estado de AD0. Si no se especifica, 0x68 estará implicito
MPU6050 sensor;

// Valores RAW (sin procesar) del giroscopio en el eje z.
int16_t gx, gy, gz;
int16_t ang_z;
int16_t ang_zr;
int16_t ang_gyr_z;

long tiempo_prev;
float dt;

void printTab()
{
  Serial.print(F("\t"));
}


WiFiClient espClient;
PubSubClient client(espClient);

unsigned long lastMsg = 0;
#define MSG_BUFFER_SIZE  (50)
char msg1[MSG_BUFFER_SIZE];



void setup_wifi() {

  delay(10);
  // We start by connecting to a WiFi network
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
```

```
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  randomSeed(micros());

  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}

void callback(char* topic, byte* payload, unsigned int length) {
  Serial.print("Message arrived [");
  Serial.print(topic);
  Serial.print("] ");
  for (int i = 0; i < length; i++) {
    Serial.print((char)payload[i]);
  }
  Serial.println();

  // Switch on the LED if an 1 was received as first character
  if ((char)payload[0] == '1') {
    digitalWrite(BUILTIN_LED, LOW);   // Turn the LED on (Note that LOW is
the voltage level
    // but actually the LED is on; this is because
    // it is active low on the ESP-01)
  } else {
    digitalWrite(BUILTIN_LED, HIGH);  // Turn the LED off by making the
voltage HIGH
  }

}

void reconnect() {
  // Loop until we're reconnected
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    // Create a random client ID
    String clientId = "ESP8266Client-";
    clientId += String(random(0xffff), HEX);
    // Attempt to connect
    if (client.connect(clientId.c_str())) {
      Serial.println("connected");
      // Once connected, publish an announcement...
      client.publish("Casa/Habitación/AnguloPuerta", "Enviando el primer
mensaje");
      // ... and resubscribe
      client.subscribe("inTopic");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      // Wait 5 seconds before retrying
      delay(5000);
    }
  }
```

```
}

void setup() {
  Serial.begin(9600);
  Wire.begin();
  sensor.initialize();     //Iniciando el sensor

  if (sensor.testConnection()) Serial.println("Sensor iniciado
correctamente");
  else Serial.println("Error al iniciar el sensor");

  sensor.setXGyroOffset(-16);
  sensor.setYGyroOffset(72);
  sensor.setZGyroOffset(-221);

  setup_wifi();
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
}

void loop() {

  if (!client.connected()) {
    reconnect();
  }
  client.loop();

// Leer la velocidad angular en el eje z.

   sensor.getRotation(&gx, &gy, &gz);

  // Se aplica el ratio de conversión a la velocidad angular leida.

  ang_gyr_z=gz*(250.0/32768.0);

  //Serial.print("La velocidad angular es: ");
  //Serial.print(ang_gyr_z);
  //Serial.print("\xc2\xb0");
  //Serial.println("/s");

  //Se calcula el ángulo en z.

  dt=(millis()-tiempo_prev)/1000;
  tiempo_prev=millis();

  //Serial.print("Dt: ");
  //Serial.print(dt);
  //Serial.println(" s");
  //Serial.print("Tiempo previo: ");
  //Serial.print(tiempo_prev);
  //Serial.println(" ms");

  ang_z=ang_z+ang_gyr_z*dt;
  ang_zr=ang_z*-1;
  //Mostrar el valor del ángulo.


   //Serial.print("Ángulo de apertura : ");
   //Serial.print(ang_z);
   //Serial.println("\xc2\xb0");
```

```
    //++value;
    snprintf (msg1, MSG_BUFFER_SIZE, "%d", ang_zr);

   // Serial.print("Publish message: ");
    Serial.println(msg1);
    client.publish("Casa/Habitación/AnguloPuerta", msg1);

    delay(1000);
}
```

En la placa de desarrollo situada en la ventana, el proyecto alojado es el siguiente:

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include "I2Cdev.h"
#include "Wire.h"
#include "MPU6050.h"


// Update these with values suitable for your network.

const char* ssid = "MIWIFI_2G_Hmmx";
const char* password = "EPpubggN";
const char* mqtt_server = "192.168.1.163";

// La dirección del MPU6050 puede ser 0x68 o 0x69, dependiendo
// del estado de AD0. Si no se especifica, 0x68 estará implicito
MPU6050 sensor;

// Valores RAW (sin procesar) del giroscopio en el eje z.
int16_t gx, gy, gz;
int16_t ang_z;
int16_t ang_gyr_z;

long tiempo_prev;
float dt;

void printTab()
{
  Serial.print(F("\t"));
}


WiFiClient espClient;
PubSubClient client(espClient);

unsigned long lastMsg = 0;
#define MSG_BUFFER_SIZE  (50)
char msg1[MSG_BUFFER_SIZE];



void setup_wifi() {

  delay(10);
  // We start by connecting to a WiFi network
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.mode(WIFI_STA);
```

```
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  randomSeed(micros());

  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}

void callback(char* topic, byte* payload, unsigned int length) {
  Serial.print("Message arrived [");
  Serial.print(topic);
  Serial.print("] ");
  for (int i = 0; i < length; i++) {
    Serial.print((char)payload[i]);
  }
  Serial.println();

  // Switch on the LED if an 1 was received as first character
  if ((char)payload[0] == '1') {
    digitalWrite(BUILTIN_LED, LOW);   // Turn the LED on (Note that LOW is
the voltage level
    // but actually the LED is on; this is because
    // it is active low on the ESP-01)
  } else {
    digitalWrite(BUILTIN_LED, HIGH);  // Turn the LED off by making the
voltage HIGH
  }

}

void reconnect() {
  // Loop until we're reconnected
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    // Create a random client ID
    String clientId = "ESP8266Client-";
    clientId += String(random(0xffff), HEX);
    // Attempt to connect
    if (client.connect(clientId.c_str())) {
      Serial.println("connected");
      // Once connected, publish an announcement...
      client.publish("Casa/Habitación/AnguloVentana", "Enviando el primer
mensaje");
      // ... and resubscribe
      client.subscribe("inTopic");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      // Wait 5 seconds before retrying
      delay(5000);
    }
  }
}
```

```
void setup() {
  Serial.begin(9600);
  Wire.begin();
  sensor.initialize();     //Iniciando el sensor

  if (sensor.testConnection()) Serial.println("Sensor iniciado
correctamente");
  else Serial.println("Error al iniciar el sensor");

  sensor.setXGyroOffset(46);
  sensor.setYGyroOffset(28);
  sensor.setZGyroOffset(-43);

  setup_wifi();
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
}

void loop() {

  if (!client.connected()) {
    reconnect();
  }
  client.loop();

// Leer la velocidad angular en el eje z.

   sensor.getRotation(&gx, &gy, &gz);

  // Se aplica el ratio de conversión a la velocidad angular leida.

  ang_gyr_z=gz*(250.0/32768.0);

  //Serial.print("La velocidad angular es: ");
  //Serial.print(ang_gyr_z);
  //Serial.print("\xc2\xb0");
  //Serial.println("/s");

  //Se calcula el ángulo en z.

  dt=(millis()-tiempo_prev)/1000;
  tiempo_prev=millis();

  //Serial.print("Dt: ");
  //Serial.print(dt);
  //Serial.println(" s");
  //Serial.print("Tiempo previo: ");
  //Serial.print(tiempo_prev);
  //Serial.println(" ms");

  ang_z=ang_z+ang_gyr_z*dt;

  //Mostrar el valor del ángulo.


   //Serial.print("Ángulo de apertura : ");
   //Serial.print(ang_z);
   //Serial.println("\xc2\xb0");

    //++value;
```

```
    //snprintf (msg1, MSG_BUFFER_SIZE, "El ángulo de apertura es: %d",
ang_z);
    snprintf (msg1, MSG_BUFFER_SIZE, "%ld", ang_z);

   // Serial.print("Publish message: ");
    Serial.println(msg1);
    client.publish("Casa/Habitación/AnguloVentana", msg1);

    delay(1000);
}
```

7.7.2 Proyecto alojado en la ESP32.

El proyecto alojado en la placa de desarrollo ESP32 es el siguiente:

```
//Librerías.

#include <WiFi.h>
#include <PubSubClient.h>

//Datos relacionados con la red local.

const char* ssid = "MIWIFI_2G_Hmmx";
const char* password = "EPpubggN";
const char* mqtt_server = "192.168.1.163";

//Datos relacionados con MQTT (Publicación/Suscripción)

WiFiClient espClient;
PubSubClient client(espClient);
unsigned long lastMsg = 0;
#define MSG_BUFFER_SIZE  (50) //Defines el tamaño del buffer.
char msg[MSG_BUFFER_SIZE]; //Se crea un array tipo char con el tamaño del
buffer especificado en la línea anterior.


//Datos necesarios para el correcto funcionamiento del sensor de gas
SEN0159.

#define         MG_PIN                       (39)     //Se define el pin
análogo a utilizar.
#define         BOOL_PIN                     (2)
#define         DC_GAIN                      (8.5)    //Se define la
ganancia DC del amplificador.

#define         READ_SAMPLE_INTERVAL         (100)     //Se define cuantas
muestras se va a utilizar en un funcionamiento normal.
#define         READ_SAMPLE_TIMES            (5)      //Define el intervalo
de tiempo (milisegundos) que hay entre cada muestra en un funcionamiento
normal.


#define         ZERO_POINT_VOLTAGE           (0.297) //Define la salida del
sensor en voltios cuando la concentración de CO2 es 400ppm
#define         REACTION_VOLTGAE             (0.030) //Define la caida de
voltaje cuando el nivel de CO2 en el sensor sube a 1000ppm

float           CO2Curve[3]  =  {2.602,ZERO_POINT_VOLTAGE,(REACTION_VOLTGAE
/(2.602-3))};
                                                          //Se forma una
línea con dos puntos que se aproxima a la curva original.
```

```
                                                                //Formato de los
datos:{ x, y, slope}; Punto 1: (lg400, 0.324), Punto 2: (lg4000, 0.280)
                                                                //slope = (
reaction voltage ) / (log400 –log1000)

//Conexión WI-FI.

void setup_wifi() {

  delay(10);
  //We start by connecting to a WiFi network
  Serial.println();
  Serial.print("Conectando a ");
  Serial.println(ssid);

  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}

//Función que permite la conexión con el  broker MQTT.

void reconnect() {
  // Loop until we're reconnected
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    // Create a random client ID
    String clientId = "ESP8266Client-";
    clientId += String(random(0xffff), HEX);
    // Attempt to connect
    if (client.connect(clientId.c_str())) {
      Serial.println("Connected");
     //Once connected, publish an announcement...
     //client.publish("Casa/Habitación/NivelGas", "Enviando el primer
mensaje: ");
     // ... and resubscribe
     // client.subscribe("inTopic");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      // Wait 5 seconds before retrying
      delay(5000);
    }
  }
}


void setup()
{
    Serial.begin(9600);                                 //Inicio del montitor
serial, baudrate = 9600bps
```

```
    setup_wifi();

    pinMode(BOOL_PIN, INPUT);                           //Establecer el pin de
entrada
    digitalWrite(BOOL_PIN, HIGH);                       //Encender las
resistencias pullup

    client.setServer(mqtt_server, 1883);
}

void loop()
{
   if (!client.connected()) {
    reconnect();
   }
   client.loop();

    int percentage;
    float volts;

    volts = MGRead(MG_PIN);
    Serial.print( "SEN0159:" );
    Serial.print(volts);
    Serial.print( "V           " );

    percentage = MGGetPercentage(volts,CO2Curve);
    Serial.print("CO2:");
    if (percentage == -1) {
        Serial.print( "<400" );
    } else {
        Serial.print(percentage);
        snprintf (msg, MSG_BUFFER_SIZE,"%ld", percentage);
        Serial.print("Mensaje publicado: ");
        Serial.print(msg);
        client.publish("Casa/Habitación/NivelCO2",msg);
    }

    Serial.print( "ppm" );
    Serial.print("\n");

    if (digitalRead(BOOL_PIN) ){
        Serial.print( "=====BOOL is HIGH======" );
    } else {
        Serial.print( "=====BOOL is LOW======" );
    }

    Serial.print("\n");

    delay(500);
}


//Esta función lee la salida del SEN-000007

float MGRead(int mg_pin)
{
    int i;
    float v=0;

    for (i=0;i<READ_SAMPLE_TIMES;i++) {
```

```
        v += analogRead(mg_pin);
        delay(READ_SAMPLE_INTERVAL);
    }
    v = (v/READ_SAMPLE_TIMES) *5/4095 ;
    return v;
}


// Una potencia de 10 es utilizada para convertir el valor en un valor no
logarítmico.

int  MGGetPercentage(float volts, float *pcurve)
{
   if ((volts/DC_GAIN )>=ZERO_POINT_VOLTAGE) {
      return -1;
   } else {
      return pow(10, ((volts/DC_GAIN)-pcurve[1])/pcurve[2]+pcurve[0]);
   }
}
```

7.7.3 Proyecto Python alojado en la Raspberry Pi.

El proyecto alojado en la Raspberry Pi es el siguiente:

```
import paho.mqtt.client as mqtt
import paho.mqtt.publish as publish
import pytz
import datetime
import pymongo

broker_address = "192.168.1.163"
broker_port = 1883
topic_sub0 = "Casa/Habitación/NivelCO2"
topic_sub1 = "Casa/Habitación/AnguloVentana"
topic_sub2 = "Casa/Habitación/AnguloPuerta"
topic_pub0 = "Recogida/CO2"
topic_pub1 = "Recogida/AnguloV"
topic_pub2 = "Recogida/AnguloP"

lista = []
# MongoDB

uri =
'mongodb://pedro_jose:tyKoRJJenqy79xBv@asia.ujaen.es:8048/?authSource=pedro
_jose_tfm&athMechanism=SCRAM-SHA-256'
myclient = pymongo.MongoClient(uri)
mydb = myclient["pedro_jose_tfm"]
def on_message(client, userdata, message):
  mensaje = str(message.payload.decode("utf-8"))
  print("Mensaje recibido=", str(message.payload.decode("utf-8")))
  print("Topic=", message.topic)
  print("Nivel de calidad [0|1|2]=", message.qos)
  print("Flag de retencion =", message.retain)
  timestamp = datetime.datetime.now(pytz.timezone('Europe/Madrid'))
  timestamp_str = timestamp.strftime("%d/%m/%Y, %H:%M:%S")
  print("Timestamp=", timestamp_str)
  if message.topic == "Casa/Habitación/NivelCO2":
    mycol = mydb["NivelCO2"]
    publish.single(topic_pub0, mensaje, hostname="192.168.1.163")
    print("Mensajes de las medidas recogidas publicados")
    list = [{'medidaCO2': mensaje, 'timestamp': timestamp_str}]
    mycol.insert_many(list)
```

```
  if message.topic == "Casa/Habitación/AnguloVentana":
     mycol = mydb["AnguloVentana"]
     publish.single(topic_pub1, mensaje, hostname="192.168.1.163")
     print("Mensajes de las medidas recogidas publicados")
     list = [{'medidaAnguloV': mensaje, 'timestamp': timestamp_str}]
     mycol.insert_many(list)
  if message.topic == "Casa/Habitación/AnguloPuerta":
         mycol = mydb["AnguloPuerta"]
         publish.single(topic_pub2, mensaje, hostname="192.168.1.163")
         print("Mensajes de las medidas recogidas publicados")
         list = [{'medidaAnguloP': mensaje, 'timestamp': timestamp_str}]
         mycol.insert_many(list)
while(True):
  try:
   client = mqtt.Client('Cliente1')
   client.on_message = on_message
   client.connect(broker_address, broker_port, 60)
   client.subscribe([(topic_sub0, 0), (topic_sub1, 0), (topic_sub2, 0)])  #
Subscription a topic_sub
   client.loop_forever()
  except Exception as e:
   print('e')
```

8. Bibliografía.

1. “Índice nacional de calidad del aire”. Gobierno de España. Año: 2022. Dirección url: https://funcionpublica.hacienda.gob.es/dam/es/portalsefp/gobernanza-publica/calidad/reconocimiento/premios/premiosXV/SE_Medio_Ambiente_Indice_Nacional_Calidad_Aire_Espania.pdf.pdf
2. “Is CO_2 an Indoor Pollutant? Direct Effects of Low-to-Moderate CO_2 Concentrations on Human Decision-Making Performance”. Usha Satish, Mark J.Mendell, Krishnamurthy Shekhar, Toshifumi Hotchi, Douglas Sullivan, Siegfriend Streufert y William J.Fisk. Año: 2012. Dirección url: https://ehp.niehs.nih.gov/doi/10.1289/ehp.1104789
3. “Airborne transmisión of respiratory viruses”. Chia C.Wang, Kimberly A.Prather, Josué sznitman, Jose L.Jimenez, Seema S.Lakdawala, Zeynep Tufekci y Linsey C. Marr. Año: 2021. Dirección url: https://www.science.org/doi/10.1126/science.abd9149
4. “Evaluación del riesgo de la transmisión de SARS-CoV-2 mediante aerosoles. Medidas de prevención y recomendaciones”. Gobierno de España. Año: 2020. Dirección url: https://www.sanidad.gob.es/profesionales/saludPublica/ccayes/alertasActual/nCov/documentos/COVID19_Aerosoles.pdf
5. “Correcting the IoT history”. Chetan Sharma. Año: 2016. Dirección url: http://www.chetansharma.com/correcting-the-iot-history/
6. “Internet of Things.”. Oxford Dictionaries. Año: 2015. Dirección url: https://www.oxfordlearnersdictionaries.com/definition/english/internet-of-things?q=internet+of+things
7. “Overview of the Internet of Things”. ITU. Año: 2012. Dirección url: https://www.itu.int/ITU-T/recommendations/rec.aspx?rec=Y.2060
8. “Selected Pollutants”. World health Organization. Año: 2010. Dirección url:https://www.who.int/publications/i/item/9789289002134
9. “Indoor Air Quality”. Amy Keller. Año: 2021. Dirección url:https://www.consumernotice.org/environmental/indoor-air-quality/#:~:text=Indoor%20Air%20Quality%201%20Common%20Pollutants%20Indoor%20air,Smoke%20...%208%20Carbon%20Monoxide%20...%20M%C3%A1s%20elementos

10. "An electronic nose based on solid state sensor array for low-cost indoor air quality monitoring applications". Zampolli, I.Elmi, F.Ahmed, M.Passini, G.C.Cardininali, S.Nicoletti y L.Dori. Año: 2004. Dirección url: https://www.sciencedirect.com/science/article/pii/S0925400504000784?casa_token=MIbhmhXwUDEAAAAA:a0AwQxSfaJSLu6wzX-Hl7l892W6Wwwa_eqX3stEk732ixezJ6-VH2O4wMpTWnlhCcLd6aQamLvMM
11. "An integrated Sensing Systems for Real-Time Indoor Air Quality Monitoring". Jung-Yoon Kim, Chao-Hsien Chu, Sang-Moon Shin. Año: 2014. Dirección url: https://ieeexplore.ieee.org/abstract/document/6907986?casa_token=YyuFmkg6P2sAAAAA:qLV12--QsXozy_dgfGB1nacUIcESzOBC1mNqxCpWqFrd_hZS-pk28z3MOF4bLD77XATdxBHSZzpM
12. "An Intelligent Wireless Sensing and Control System to Improve Indoor Air Quality: Monitoring, Prediction, and Preaction". Tsang-Chu Yu y Chung-Chih Lin. Año: 2015. Dirección url:https://journals.sagepub.com/doi/pdf/10.1155/2015/140978
13. "Indoor Air Quality Analysis Using Deep Learning with Sensor Data". Jaehyun Ahn, Dongil Shin, Kyuho Kim y Jihoon Yang. Año: 2017. Dirección url:https://www.mdpi.com/1424-8220/17/11/2476
14. "Wireless Sensor Network Combined with Cloud Computing for Air Quality Monitoring". Patricia Arroyo, José Luis Herrero, José Ignacio Suárez y Jesus Lozano. Año: 2019. Dirección url:https://www.mdpi.com/1424-8220/19/3/691
15. "Air quality monitoring using mobile microscopy and machine learning". Yi-Chen Wu, Ashutosh Shiledar, Yi-Cheng Li, Jeffrey Wong, Steve Feng, Xuan Chen, Christine Chen, Kevin Jin, Saba Janamian, Zhe Yang, Zachary Scott Ballard, Zoltán Gorocs, Alborz Feizi y Aydogan Ozcan. Año: 2017. Dirección url:https://www.nature.com/articles/lsa201746
16. "Field evaluation of a low-cost indoor air quality monitor to quantify exposure to pollutants in residential environment". Moreno-Rangel, Alejandro, Sharpe, Tim, Masau, Filbert, McGill y Grainne. Año: 2018. Dirección url:https://strathprints.strath.ac.uk/71564/
17. "Requisito funcional". Wikipedia. Dirección url: https://es.wikipedia.org/wiki/Requisito_funcional
18. "Diagrama de Gantt". Paula Rodó. Dirección url: https://economipedia.com/definiciones/diagrama-de-gantt.html

19. “Metodologías de desarrollo de software: características, tipos, ventajes y desventajes”. Dirección url: https://blog.gitnux.com/es/metodologias-de-desarrollo-de-software/
20. “Modelo evolutivo de desarrollo en espiral”. Gabriel Mancuzo. Dirección url: 1. “Metodologías de desarrollo de software: características, tipos, ventajes y desventajes”. Dirección url: https://blog.gitnux.com/es/metodologias-de-desarrollo-de-software/
21. “Modelo de cascada”. Anastasia Stsepanets. Dirección url: https://blog.ganttpro.com/es/metodologia-de-cascada/
22. “Modelo incremental”. Anna Pérez. Dirección url: https://www.obsbusiness.school/blog/caracteristicas-y-fases-del-modelo-incremental
23. “SEN0159”. DFROBOT. Dirección url: https://wiki.dfrobot.com/CO2_Sensor_SKU_SEN0159
24. “¿Qué es MQTT? Su importancia como protocolo MQTT”. Luis LLamas. Dirección url: https://www.luisllamas.es/que-es-mqtt-su-importancia-como-protocolo-iot/
25. “Qué son y cómo usar los topics en MQTT correctamente”. Luis LLamas. Dirección url: https://www.luisllamas.es/que-son-y-como-usar-los-topics-en-mqtt-correctamente/
26. “Bases de datos relacionales”. IBM. Dirección url: https://www.ibm.com/mx-es/topics/relational-databases
27. “Bases de datos no relacionales”. IBM. Dirección url: https://www.ibm.com/mx-es/topics/nosql-databases