Escuela Politécnica Superior de Jaén

Máster Universitario en Ingeniería Informática

UNIVERSIDAD DE JAÉN

Escuela Politécnica Superior (Jaén)

MONITORIZACIÓN DE AMBIENTES CON DISPOSITIVOS DE ÚLTIMA GENERACIÓN

Alumna: Barbero Rodríguez, María de la Paz

Tutor: José Luis López Ruiz

Cotutor: Alicia Montoro Lendínez

Dpto.: Departamento de Informática

Julio 2022

Contenido

Índice de Figuras

Índice de tablas

1. Descripción del trabajo.

1.1. Introducción.

La enfermedad por coronavirus ha sido desde su aparición y sigue siendo, uno de los principales problemas de salud a nivel mundial, por lo que también es uno de los principales focos de atención mediática debido a las terribles consecuencias que conlleva. Desde que comenzó la pandemia, hemos sido bombardeados con información relativa a la prevención de la infección de esta enfermedad con toda clase de acciones y, casi rituales, para evitar el contagio. En un primer momento, se consideró el contagio directo a través de las superficies o las gotículas expulsadas, como la principal vía de transmisión, pero, gracias a la experiencia y a las posteriores investigaciones llevadas a cabo, se ha considerado, cada vez con más fuerza, la transmisión del virus mediante la inhalación de aerosoles que quedan suspendidos en el aire durante largos períodos de tiempo y que, por ello, podemos inhalar y exhalar con total facilidad. Estos aerosoles se acumulan, sobre todo, en espacios interiores y cerrados, haciendo que, con el paso del tiempo, la cantidad de partículas sea mayor por lo que en el caso de ser producidas por personas infectadas, la carga vírica también aumente considerablemente [1].

Por este motivo, cada vez se pone mayor énfasis en la necesidad de la ventilación de los espacios cerrados para reducir la posible carga viral y este hecho se manifiesta, en la importancia concedida desde las grandes instituciones, como el Gobierno de España, que pone a disposición de la ciudadanía información relativa a concentración de partículas, sistemas de ventilación o distancias de seguridad, referidas tanto al ámbito doméstico [2] como al ámbito de carácter público [3].

Al margen de esta enfermedad, hay estudios que evidencian los efectos que sufren las personas debido a la exposición en ambientes con diferentes niveles de partículas de dióxido de carbono (en adelante CO_2), concluyendo que, a concentraciones mayores, los efectos nocivos para la salud eran también mayores. Estos efectos hacen referencia al rendimiento personal, por ejemplo, en la toma de decisiones mediante la realización de diversas tareas, según el

estudio de Satish et al. [4], además de afectar al estado físico (dolores de cabeza, mareos, etc.) o fisiológico (aumento de la presión sanguínea). Dado que las personas exhalamos CO_2 al respirar, los niveles de este gas van aumentando en función del tiempo de estancia en habitaciones cerradas sin ventilación, por lo que, se vuelve a hacer visible la importancia de la ventilación en estas circunstancias. Sin embargo, debido a la naturaleza de ciertas actividades como es el caso de las clases de docencia presenciales, es inevitable la congregación de un alto número de personas en estancias en las que no sería viable tener las ventanas abiertas de forma permanente debido al gasto que se generaría en materia de calefacción, por ejemplo.

En este punto se puede deducir la doble ventaja de medir los niveles de CO_2 de las estancias, por una parte, controlar la calidad del aire en cuanto al ya perjudicial CO_2 y, por otra parte, estimar el posible nivel de carga viral que se encuentra en el ambiente mediante la relación proporcional entre la cantidad de aire exhalado (que puede contener el virus) y el incremento de CO_2 que producen las personas al exhalar ese aire, pues, a mayor cantidad de CO_2, mayor cantidad de aire “ya respirado por otra persona” que puede contener el virus. La consecuencia de los datos registrados es que gracias a ellos se puede considerar cuándo ventilar la habitación y cómo afecta esta ventilación a los niveles de CO_2 que nos indican la calidad del aire.

En esta situación es importante hacer uso de las nuevas tecnologías con las que cualquier elemento puede ser útil para obtener datos, analizarlos y aprovechar dicha información para un posterior conocimiento relevante.

Por tanto, el propósito de este trabajo es abordar el problema real actual sobre la calidad del aire, así como proponer una posible solución al mismo mediante la monitorización de ambientes inteligentes de la mano del Internet de las Cosas (IoT). A lo largo de este documento, se explica en su totalidad en qué consiste y cómo se logra dicha solución.

1.2. Objetivos del trabajo.

El objetivo principal del trabajo es el desarrollo de un sistema que permita conocer en cada momento el estado de la calidad del aire, centrándonos, concretamente, en concentraciones de CO_2. De esta forma, se puede obtener información del estado de una estancia tomando en cuenta diversas variables, relacionadas con la propia estancia, para así ver cómo evoluciona la calidad del aire en ciertas circunstancias.

Los objetivos específicos del TFM son:

1. Definir el conjunto de procesos necesarios para desarrollar el sistema de monitorización de ambientes inteligentes.
2. Desarrollar los procesos asociados para el almacenamiento de la información necesaria.
3. Desplegar un sistema de dispositivos inteligentes que permita la monitorización de ambientes inteligentes.
4. Realizar los manuales asociados.
5. Redactar una memoria que recoja todo el trabajo desarrollado, así como los manuales de despliegue e instalación.

1.3. Estado del arte.

Este trabajo abarca dos campos diferentes muy relacionados entre sí, como son el Internet de las Cosas (IoT) y la monitorización de ambientes.

En el primer campo, la denominación de IoT nace en 1999 de la mano del ingeniero informático Kevin Ashton quien trabajaba en el ámbito de la identificación de las cosas por radiofrecuencia (RFID) , y afirmaba que había que reconsiderar el papel de las ‘cosas’ en el mundo físico en el que vivimos y con los avances computacionales que existían [5]. De este término han surgido múltiples definiciones durante los últimos años, llegando a ser una de las palabras que más escuchamos relacionadas con el mundo de la informática, hemos vivido un crecimiento exponencial del IoT, aunque el concepto en sí, a

veces, queda un poco difuso y se confunde con otros conceptos que se verán más adelante.

Según la compañía Gartner, el IoT es la red de objetos físicos que contienen tecnología incorporada para comunicarse y percibir o interactuar con sus estados internos o el entorno externo [6]. En cambio, otra de las grandes compañías como Cisco define IoT como un hito en el tiempo, concretamente, el momento en el que hubo más cosas u objetos conectados a Internet que personas, hecho que ocurrió entre 2008 y 2009 [7]. Ambas definiciones, aunque desde perspectivas diferentes, confluyen en mencionar la idea de cosas conectadas a la red, por tanto, esas "cosas" deben contener una tecnología incorporada que les permita la conectividad entre ellas y/o con Internet, además de poder generar (o no) datos para poder transmitirlos y poder realizar alguna acción (o no) en función de los datos que recibe o detecta.

El futuro de esta disciplina es el Internet de Todas las Cosas (IoE, Internet of Everything), que se define como la conexión en red de personas, procesos, datos y cosas y se beneficia del valor que crea esta gran conexión a medida que todo se pone en línea. Esta definición da un paso más, ya que, mientras IoT es una transición tecnológica, IoE comprende muchas transiciones tecnológicas, incluyendo IoT [8].

Como se puede pensar, IoT (y, por supuesto, IoE) pueden aplicarse en numerosos campos, tanto a nivel doméstico como a nivel empresarial, industrial, o en el sector público, en sanidad y en educación.

Una aplicación muy relevante y relacionada con este trabajo que tiene el IoT son los ambientes inteligentes, que podemos definir como entornos electrónicos sensibles y receptivos a la presencia de un usuario, que pasan inadvertidos en el entorno hasta que sólo la interfaz de usuario permanece perceptible para los usuarios. Este paradigma (Inteligencia Ambiental, AmI) se basa en la computación pervasiva o ubicua, el perfilado, la conciencia del contexto y el diseño de la interacción centrada en las personas [9]. Un ejemplo de un ambiente inteligente es el Living Lab de la Universidad de Jaén, un

laboratorio repleto de sensores integrados en diferentes lugares que es utilizado en múltiples aplicaciones y proyectos [10].

Volviendo a otra parte de nuestro caso de estudio, la monitorización de gases (CO_2), un sensor de gas es un sensor químico compuesto por un transductor y una capa activa que permite convertir la información química en algún tipo de señal electrónica, como un cambio de voltaje. Estos sensores tienen algunas características propias, como la sensibilidad, la selectividad entre varios gases, el límite de detección, el tiempo de respuesta y el tiempo de recuperación, características que permiten determinar su rendimiento atendiendo a diferentes perspectivas.

Haciendo una revisión bibliográfica, podemos encontrar varios tipos de sensores de gas, entre los que destacan, según Z. Yunusa [11]:

- **Sensores catalíticos**. Se utilizan para la detección de gases combustibles y se basan en el proceso de combustión catalítica, donde, gracias a una reacción química, se necesita de una menor temperatura para la ignición de los gases combustibles y, con un puente de Wheatstone, se puede medir la salida de datos de este sensor. Son sensores simples para detectar la inflamabilidad de los gases, aunque necesitan aire u oxígeno para funcionar.
- **Sensores de conductividad térmica**. Suelen emplearse para detectar gases con conductividades térmicas elevadas, superiores a la del aire, como el hidrógeno y el metano. El principio de su funcionamiento se basa en la pérdida de calor medida, gracias a una termorresistencia, desde un cuerpo más caliente hacia el elemento frío a través de la conductividad térmica. Son sensores simples pero robustos, sin embargo, necesitan un hilo calentador.
- **Sensores electroquímicos**. Este tipo de sensores permite que los gases se difundan a través de una membrana porosa hasta un electrodo donde se reducen u oxidan, dependiendo del gas en cuestión. Los sensores electroquímicos funcionan reaccionando con un gas objetivo y produciendo una señal eléctrica que es proporcional a la concentración

de dicho gas. Pueden detectar una gran cantidad de gases en concentraciones relativamente bajas, aunque los modos de fallo no se revelan a menos que se utilice una técnica de supervisión avanzada.

- **Sensores ópticos**. Este tipo de sensores utiliza la emisión de la luz a una longitud de onda específica, lo que produce luminiscencia (fluorescencia o fosforescencia), que es capaz de ser medida por el sensor y calcular así la concentración del gas. El principal inconveniente que presenta este tipo de sensores es que se ve afectado por la luz ambiente, pero presenta otras ventajas, como que no les influyen las interferencias electromagnéticas y que el área de detección es amplia.
- **Sensores infrarrojos**. Estos sensores son un tipo de los mencionados anteriormente (sensores ópticos), en los que la luz es infrarroja. Consisten en un detector que convierte la energía de la radiación electromagnética en señales eléctricas. Dentro de este grupo de sensores podemos destacar los sensores NDIR (Non-Dispersive Infrared), muy utilizados en los detectores de CO_2 comercializados en los últimos tiempos.
- **Sensores semiconductores**. Son dispositivos formados por óxidos metálicos calentados que se utilizan para medir la concentración de un gas objetivo mediante la medición de la resistencia eléctrica del dispositivo. Normalmente se trata de una capa de óxido de estaño caliente que, con el contacto con el gas objetivo provoca un cambio en la resistencia eléctrica de dicho material, que podemos medir para determinar la concentración del gas. Los sensores MQ son un ejemplo de sensores semiconductores. Son robustos y funcionan bien con condiciones de alta humedad, aunque son susceptibles a cambios medioambientales y su respuesta no lineal hace que la complejidad aumente.
- **Sensores de onda acústica o sensores de onda acústica de superficie (SAW)**. Deben su nombre a su mecanismo de detección, que es una onda mecánica o acústica. Funcionan con transductores que transforman la señal eléctrica en ondas mecánicas y viceversa, de forma

que, debido al efecto piezoeléctrico, cambian las características de propagación de las ondas cuando se exponen al gas objetivo. Estos cambios en las propiedades de la onda son proporcionales a la concentración del gas.

Según Zaynab Yunusa, los sensores más completos son los últimos mencionados de bajo coste y bajo consumo de energía para aplicaciones de detección de gases. Actualmente, y con visión futura, existe la tendencia a usar nano-materiales en la construcción de los sensores enumerados anteriormente para mejorar las características de funcionamiento y lograr así una mayor eficiencia.

Las principales aplicaciones que encontramos con estos sensores de gas hacen referencia a sistemas de seguridad contra fuga de gases (normalmente tóxicos o inflamables). Estas aplicaciones se encuentran, sobre todo, en la industria, donde, un escape de gas puede tener consecuencias desastrosas.

En la actualidad podemos encontrar aplicaciones parecidas en el mercado surgido en torno a la prevención contra el virus. Un ejemplo de ello es Aranet4, un aparato que incorpora un sensor CO_2 (NDIR) y una pantalla en la que se visualiza la concentración en partes por millón (ppm), además de la temperatura y humedad. Este sistema oscila los 200 euros en su versión básica.

1.4. Restricciones.

Las restricciones que se encuentran, a priori, son de índole temporal y están relacionados con la naturaleza del propio trabajo.

1.4.1. Temporales.

Se puede decir que, debido a las características del trabajo, para la realización de pruebas es necesario disponer de personas voluntarias que puedan permanecer de forma presencial en la estancia durante horas para ver la evolución de los valores.

1.4.2. Espacio físico.

Debido a la naturaleza del propio trabajo, es necesario un espacio para configurar, desplegar y experimentar el sistema de monitorización. Por tanto, el trabajo se ha llevado a cabo en una habitación de unos 12m^2 con la siguiente arquitectura (Figura 1).

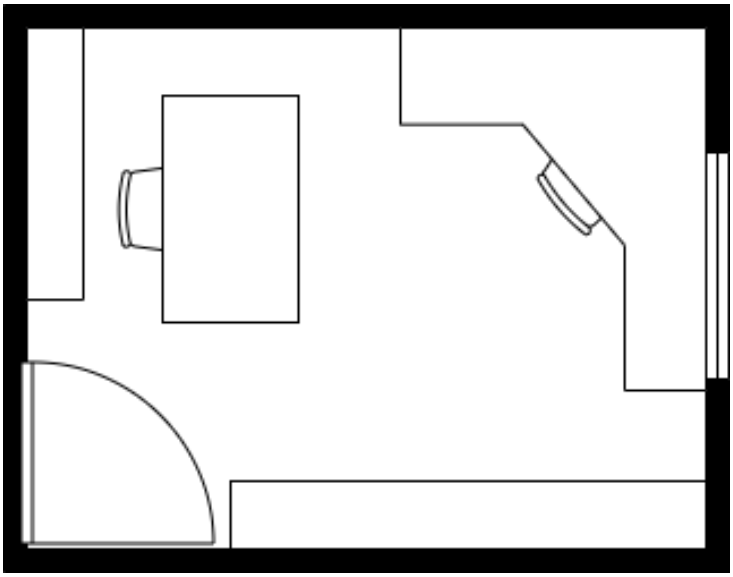


Figura 1. Plano de la habitación en la que se va a implementar el sistema.

1.4.3. Consumo eléctrico.

Una de las características más deseables del IoT suele ser el bajo consumo debido a que los sensores, a veces, tienen que trabajar con baterías para su alimentación, además de estar funcionando durante largos períodos de tiempo, por lo que un sistema con alto consumo podría suponer la inviabilidad de un proyecto o aplicación en este ámbito.

1.5. Solución propuesta.

La solución que se propone consta de la recogida de diferentes datos y la representación y visualización de los mismos. Los datos que se van a recoger son:

- Datos que cuantifican la calidad del aire, centrándose en concentraciones de CO_2.
- Datos que representan estados de los mecanismos de ventilación, la apertura/cierre de puertas y ventanas.
- Datos que indican la cantidad de personas que se encuentran en la habitación en la que se están realizando las mediciones anteriores.

Estos datos se enviarán a una aplicación que los empaquetará y los enviará a la base de datos destinada para ello para un trabajo posterior de generación de conocimiento. Por otra parte, se va a añadir un mecanismo para poder visualizar gráficamente los datos que se han medido y controlar en tiempo real las variables que definen las características de la estancia.

1.6. Metodología.

La metodología seguida en el TFM es la siguiente:

- Revisión bibliográfica de las tecnologías disponibles.
- Análisis y diseño de los procesos para el sistema de monitorización de ambientes inteligentes
- Análisis, diseño y desarrollo de los procesos utilizando los dispositivos de última generación que permitan la monitorización de ambientes inteligentes.
- Realización de los manuales asociados al prototipo.
- Generación de la memoria del trabajo realizado.

En cuanto a la metodología del desarrollo de software, entre las que se encuentra un gran abanico de posibilidades, desde las más tradicionales como en cascada, que divide el proceso de desarrollo en etapas (análisis, diseño,

implementación y pruebas), hasta las más utilizadas hoy en día, las metodologías ágiles, basadas en el desarrollo iterativo en el que los requerimientos pueden ir evolucionando durante el transcurso del proyecto. Una de las principales diferencias entre ambos enfoques es la susceptibilidad al cambio que presentan. Mientras que las tradicionales no toleran bien los cambios que puedan producirse en los requerimientos, las ágiles son presentan bastante flexibilidad en este sentido. Es por ello que son las más utilizadas a día de hoy en la mayoría de proyectos software (y no software).

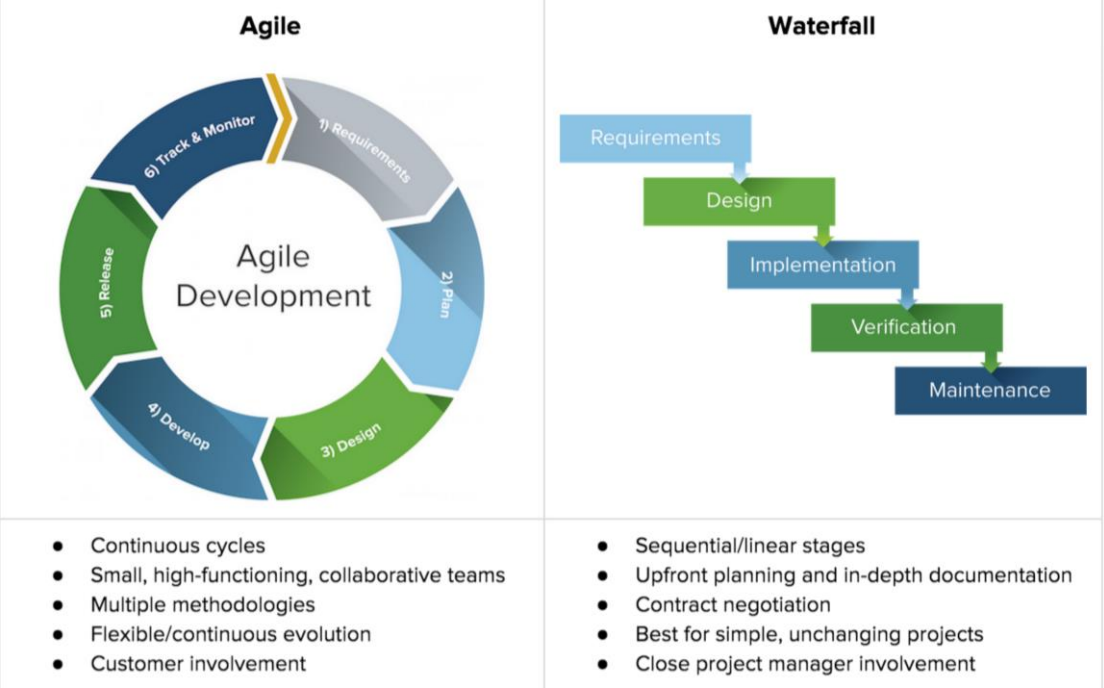


Figura 2. Metodología ágil vs metodología en cascada. Vía [12].

Por estos motivos, debido a la naturaleza cambiante, iterativa e incremental del trabajo, se va a utilizar una metodología ágil para el desarrollo del mismo. Así, se van a definir unos hitos principales en los que se abordan las tareas mencionadas en este apartado sobre los cuáles se construirán las iteraciones.

1.7. Requisitos del sistema.

Los requisitos son la base para el diseño y la implementación en la ingeniería del software. Son un conjunto de propiedades y funcionalidades que debe tener el software.

Dentro de la categoría de requisitos funcionales, se encuentran:

- El sistema debe recoger los niveles de CO_2 del ambiente como representación de la calidad del aire. Del mismo modo, debe recoger los niveles en formato Raw.
- El sistema debe conocer el estado de la puerta y la ventana, distinguiendo los estados “abierta” y “cerrada”.
- El sistema debe estimar el número de personas que se encuentran en la estancia, con la premisa de que cada persona usa un smartphone.
- El sistema debe almacenar todos los datos recogidos estimados anteriormente con la marca temporal que indica cuándo se han almacenado.
- El sistema debe poder mostrar de forma visual todos estos datos.
- Los dispositivos deben comunicarse de manera inalámbrica.
- Se debe poder gestión la visualización de los datos recogidos.

En cuanto a los requisitos no funcionales, se distinguen:

- Usabilidad. Facilidad de uso, diagramas y gráficos entendibles (visualización).
- Automatización. Una vez que el sistema está puesto en marcha, es invisible para las personas.

1.8. Estimación de esfuerzo y planificación temporal.

En este apartado se presenta un resumen de la organización del tiempo para cada una de las tareas que componen el trabajo. En la Tabla 1 se detallan las principales tareas divididas en hitos o, en este caso, incrementos.

Tarea	Duración en horas	Duración en semanas
Primer prototipo básico	**22**	**1.5**
Búsqueda información y documentación sensor MQ2	4	
Montaje placa con sensor MQ2 y botones	0,5	
Desarrollo script Arduino sensor MQ2 y ejecución de pruebas	4	
Desarrollo aplicación recogida de datos en ordenador	14	
Mecanismo apertura ventana y puerta	**56**	**3**
Buscar información, documentación y ejemplos de MMA8451	10	
Realizar montaje en placa de MMA8451 y ejecutar pruebas	15	
Buscar información, documentación y ejemplos de MPU6050	10	
Realizar montaje en placa de MPU6050 y ejecutar pruebas	17	
Desarrollar script de mecanismo de apertura con MPU6050 y ejecutar pruebas	4	
Detección de CO_2	**45**	**3**
Búsqueda información y documentación sensor MQ135	15	
Montaje placa con sensor MQ135 y ejecutar pruebas	25	
Desarrollo script Arduino con MQ135 y realizar pruebas	5	
Adaptar aplicación recogida de datos	**19**	**1.5**
Búsqueda información librerías necesarias	5	
Análisis requisitos y diseño	2	
Desarrollo aplicación	7	
Pruebas conjuntas con placas	5	
Detección personas con sniffer	**52**	**3**
Búsqueda de información sniffer	12	
Búsqueda y prueba de ejemplos de sniffer compatibles con Arduino	25	
Adaptación script al caso de uso	10	
Pruebas	5	
Plataforma IoT	**41**	**2**
Búsqueda y estudio de diferentes alternativas para elegir la plataforma	4	
Búsqueda de documentación de ThingsBoard	10	
Instalación y puesta en marcha de ThingsBoard CE	15	
Creación de dispositivos, widget y dashboard en ThingsBoard	5	
Configuración de las placas para la comunicación con ThingsBoard	2	

Pruebas	5	
Migración de la aplicación y ThingsBoard a Raspberry	**11**	**1**
Instalación de SO y puesta en marcha Raspberry	2	
Ejecución y pruebas con las placas	1	
Creación de servicio para la ejecución de la aplicación	0,5	
Instalación de ThingsBoard en Raspberry	1	
Migración de dispositivos, widget y dashboard	0,5	
Búsqueda de información de ThingsBoard Gateway	4	
Instalación y configuración de ThingsBoard Gateway en Raspberry	1	
Pruebas	1	
Experimentación	**14**	**1.5**
Realización de memoria y manuales	**44**	**1**
Total	**290**	**16.5**

Tabla 1. Desglose de hitos y tareas.

A continuación, en la Figura 3 y la Figura 4, se muestra el diagrama de Gantt que representa las diferentes fases del trabajo. El desarrollo se ha prolongado durante tanto tiempo debido a la interrupción del mismo por motivos laborales y formativos.

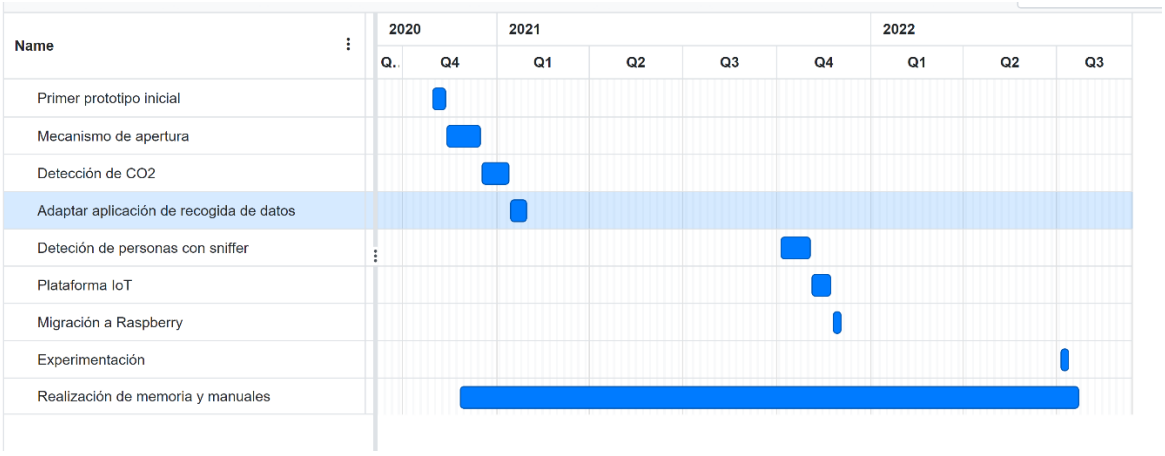
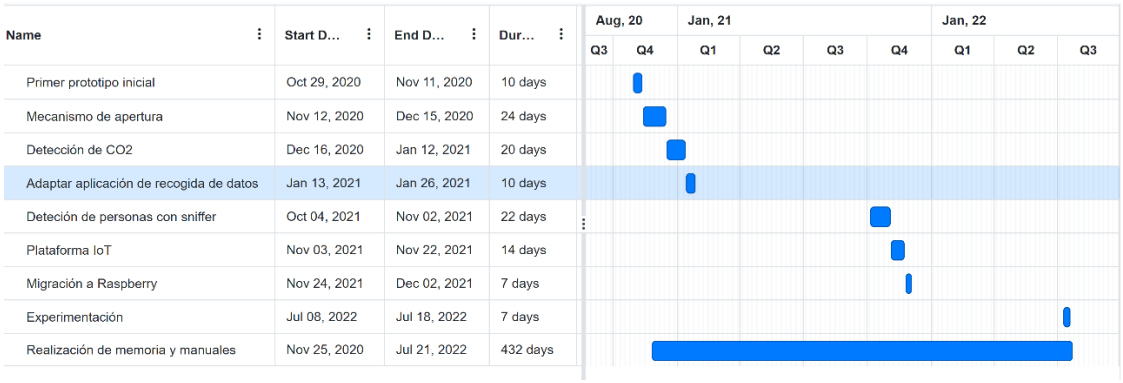


Figura 3. Diagrama de Gantt.



Name	Start D...	End D...	Dur...
Primer prototipo inicial	Oct 29, 2020	Nov 11, 2020	10 days
Mecanismo de apertura	Nov 12, 2020	Dec 15, 2020	24 days
Detección de CO2	Dec 16, 2020	Jan 12, 2021	20 days
Adaptar aplicación de recogida de datos	Jan 13, 2021	Jan 26, 2021	10 days
Deteción de personas con sniffer	Oct 04, 2021	Nov 02, 2021	22 days
Plataforma IoT	Nov 03, 2021	Nov 22, 2021	14 days
Migración a Raspberry	Nov 24, 2021	Dec 02, 2021	7 days
Experimentación	Jul 08, 2022	Jul 18, 2022	7 days
Realización de memoria y manuales	Nov 25, 2020	Jul 21, 2022	432 days

Figura 4. Diagrama de Gantt detallado.

1.9. Análisis de riesgos.

En este apartado se incluye un análisis de posibles riesgos.

Se define como riesgo, aquel evento o condición desconocida, que, en caso de ocurrir, puede tener consecuencias positivas o negativas sobre los objetivos del proyecto y sus resultados. Cualquier riesgo afecta tanto a la planificación, costes, plazos, calendario o la calidad del proyecto. Una correcta gestión de los riesgos de debe incluir aspectos relativos al entorno de desarrollo del proyecto, o dependencias de participantes y contribuciones del exterior.

El análisis de riesgos corresponde al proceso de gestión de riesgos de la planificación de proyectos. Se planifican las respuestas y se monitoriza el tratamiento de los riesgos según su probabilidad de ocurrencia e impacto.

El grado o escala en la que se van a cuantificar la probabilidad, el impacto, la gravedad y los costes sería *Muy Bajo – Bajo – Medio – Alto – Muy alto*, para que este análisis sea mucho más descriptivo.

Riesgo	Probabilidad	Impacto	Gravedad	Costes	Plan de contingencia
Experiencia insuficiente	Alto	Alto	Alto	Bajo	Ayuda de expertos (Alicia)
Sensor de gas inadecuado	Muy alto	Muy alto	Muy alto	Medio	Nuevo sensor
Sensor apertura inadecuado	Muy alto	Muy alto	Muy alto	Medio	Nuevo sensor
Retraso tiempo	Alto	Medio	Alto	Alto	Añadir un desarrollador de apoyo para agilizar las tareas.
Apagón eléctrico	Muy bajo	Muy alto	Alto	Bajo	Reinicio del sistema
Rotura Raspberry	Muy bajo	Medio	Bajo	Bajo	Exportar datos servidor para poder importarlos en otro servidor

Tabla 2. Riesgos.

1.10. Presupuesto.

El presupuesto para este proyecto se ha calculado en base a las siguientes consideraciones:

- 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 project manager y para el desarrollo del sistema de monitorización se ha estimado que basta con un programador junior.
- Por otra parte, se va a obviar el coste derivado del sueldo del empleado debido a que es un elemento que depende de muchos factores ajenos al propio proyecto.
- Arduino IDE y PyCharm no generan costes.
- Dentro del presupuesto se incluye la parte hardware necesaria para el desarrollo del sistema.
    - 1 placa Raspberry 3 – 41.95€
    - 5 placas NodeMCU - 2 v 1.0 y 2 v 3.0 (Lolin) + placa sniffer v3 – 39.95 €.

- Resistencias 10k Ohm – 0.02 € * 6 = 12cent
- Sensor MQ2 y MQ135 – mq135 6.89 €
- 2 Módulos MPU6050 – 5.99€
- 2 breadboard half+ 2 breadboard full = 4.46€*2 + 5.45€ (x2)
- Cables puente M-H pack 65 – 4.77€
- cables de alimentación 5V 2A – 9.99 € cable + enchufe (x5)
- 2 baterías – 25.45€

Concepto	Duración	Coste unitario €	Unidades	Total €
Raspberry Pi 3	Todo el desarrollo	41.95	1	41.95
Placa NodeMCU ESP8266	Todo el desarrollo	7.99	5	39.95
Resistencia 10kΩ	Todo el desarrollo	0.02	6	0.12
Sensor MQ135	Todo el desarrollo	6.89	2	13.78
Sensor MPU6050	Todo el desarrollo	5.99	2	11.98
Breadboard half	Todo el desarrollo	4.46	3	13.38
Breadboard full	Todo el desarrollo	5.45	2	10.9
Cables puente M-H, M-M	Todo el desarrollo	4.77	1 (65ud)	4.77
Cable alimentación 5V/2A	Todo el desarrollo	9.99	5	49.95
Batería portátil 26800 mAh	Todo el desarrollo	25.45	2	50.9
Total				**237.68**

Tabla 3. Desglose presupuesto.

2. Desarrollo del proyecto.

En este apartado se van a desglosar los detalles más técnicos referentes al desarrollo del sistema y a las herramientas que se han empleado para ello.

2.1. Tecnologías utilizadas.

Debido a la naturaleza tan polivalente del proyecto, se han utilizado diferentes tecnologías a lo largo del desarrollo de los componentes del mismo.

La primera consideración que se va a realizar es el protocolo de red desde la perspectiva de la capa de aplicación. Entre ellos se encuentran HTTP, MQTT o CoAP. El primero de ellos fue creado para que los documentos estuvieran disponibles en Internet. Tanto HTTP como MQTT funcionan a través de conexiones TCP y tienen una arquitectura cliente-servidor, pero MQTT permite el paso de mensajes en ambas direcciones entre clientes y servidores, mientras que los servidores HTTP sólo responden a las peticiones de los clientes.

En cuanto a CoAP, es similar a HTTP en cuanto a que, por ejemplo, utiliza un modelo de solicitud/respuesta similar, aunque funciona sobre UDP. Comparado con MQTT, es más ligero, aunque MQTT es un protocolo de comunicación de muchos a muchos para el paso de mensajes entre múltiples clientes a través de un broker central. No distingue entre productor y consumidor dejando que los clientes publiquen y que el broker decida dónde dirigir los mensajes. CoAP es, principalmente, un protocolo uno a uno para transferir información de estado entre el cliente y el servidor [13]. Debido a las características del sistema y a la experiencia obtenida, se va a optar por el uso de MQTT.

MQTT (Message Queue Telemetry Transport)

MQTT es un protocolo de mensajería para IoT basado en la publicación/suscripción. Fue creado en 1999 por Andy Stanford-Clark (IBM) y Arlen Nipper (Cirrus Link), aunque no fue hasta 2010 cuando se hizo público. Actualmente es un estándar abierto de OASIS, una organización formada por empresas y organizaciones encargada de desarrollar estándares abiertos en el ámbito de las Tecnologías de la Información. Este es uno de los motivos por los cuales MQTT se ha convertido en el protocolo de comunicación por excelencia en IoT [14]. Algunos de sus puntos fuertes son:

- Ligero y eficiente, perfecto para dispositivos con recursos escasos.
- Fiabilidad, con posibilidad de especificar calidad de servicio en tres niveles diferentes.

- Escalable a un gran número de dispositivos

Figura 5. Logo MQTT vía mqtt.org.

Por estos motivos, este protocolo es utilizado en múltiples escenarios, como la industria, la logística y transportes o en hogares inteligentes (Smart homes).

El funcionamiento de MQTT se basa en cinco elementos principales: broker, clientes, publicación, suscripción y topics. El broker es un servidor que se encarga de recibir los mensajes de los clientes y enviarlos a los clientes destinatarios. Los clientes interactúan con el broker para enviar y/o recibir mensajes. La comunicación se organiza mediante topics, que son palabras que pueden estar jerarquizadas y que hacen de buzones en los cuales los clientes pueden depositar mensajes (publicar) o recibirlos, suscribiéndose a los topics, todo esto con la mediación del broker.

El funcionamiento puede observarse con mayor claridad en la Figura 6, donde se identifican diferentes tipos de clientes: un conjunto de sensores, un administrador y uno que se encarga del almacenamiento y procesamiento de datos. En primer lugar, estos clientes se conectan al broker a través de una conexión TCP/IP simple o una conexión TLS encriptada para mensajes sensibles. A continuación, los clientes pueden suscribirse a todos los topics que deseen para obtener los mensajes publicados en ellos. Por ejemplo, los sensores obtienen datos y los publican a través de mensajes en el topic “sensor_data”, que a su vez es el topic al que está suscrito el cliente que procesa y almacena los datos. Por otra parte, estos sensores pueden recibir mensajes suscribiéndose al topic “config_change”, en el que publica el cliente encargado de la administración.

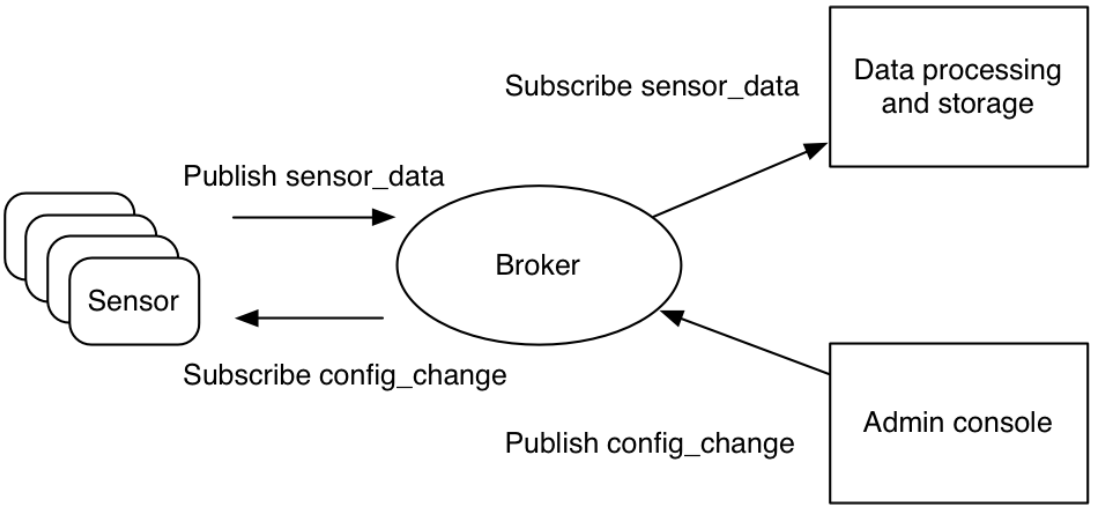


Figura 6. Ejemplo de funcionamiento de MQTT, por Michael Yuan [8].

Para incluir la comunicación mediante MQTT en este sistema se ha hecho uso de:

- Como broker en Raspberry se instala Mosquitto. Es open source y está creado por Eclipse Foundation.
- Como biblioteca para los clientes sensores, PubSubClient.
- Como biblioteca para la aplicación que recoge datos en Python, paho-mqtt.

El siguiente aspecto a considerar es la elección de los dispositivos en los que se va a implementar gran parte del sistema. Es necesario un microcontrolador con conectividad inalámbrica, capaz de adaptarse a las situaciones que se van a plantear en el trabajo. En esta área destaca la familia Arduino, en la que se encuentra la serie Nano, de tamaño reducido, en la que se reúnen las características deseables en cuanto a los requerimientos del proyecto. Por otra parte, destacan los microcontroladores un poco más asequibles que incorporan los populares módulos ESP32 y ESP8266. Ambos incluyen conectividad inalámbrica (Wifi y Bluetooth) y procesadores de 32 bits, además de pines físicos que permiten la conexión mediante protocolos como I^2C. ESP32 es el

sucesor de ESP8266 e incorpora algunas mejoras, como su procesador con doble núcleo o una mejora en las tecnologías de conectividad inalámbrica. Hay numerosas placas que incorporan estos módulos y las diferencias entre ellas son mínimas, por lo que casi que cualquiera puede servir. En este caso, debido a que se necesitan bastante placas, se ha optado por la NodeMCU ESP8266, debido en gran parte a su disponibilidad en el momento de la realización del trabajo.

Placa NodeMCU ESP8266

Para controlar el funcionamiento de los sensores y materializar sus medidas, se va a emplear la placa de desarrollo open source NodeMCU, que incorpora el SoC (system on chip) ESP8266. Esta placa tiene conectividad WiFi, un tamaño reducido y es bastante asequible, por lo que se hace ideal para las aplicaciones de IoT. Hay varias versiones de estas placas, aunque sus diferencias son menores.

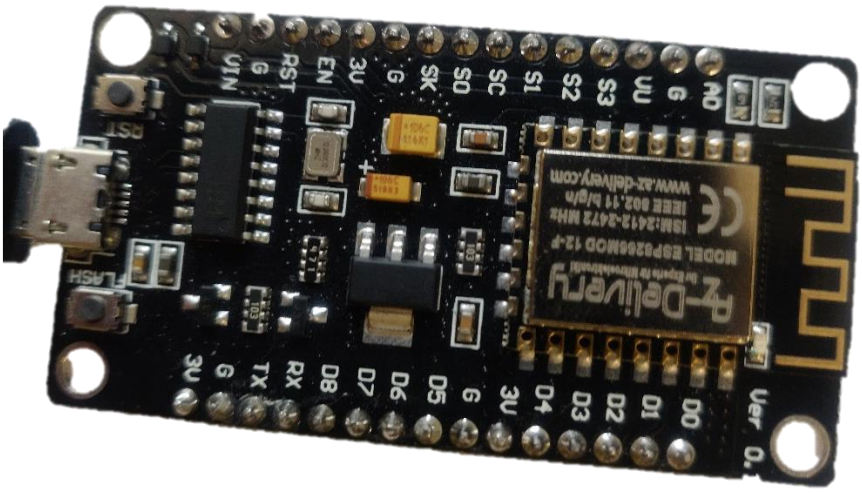


Figura 7. Placa NodeMCU.

Las conexiones con diferentes módulos y sensores se hace a través de los pines que incluye, tanto para la lectura/escritura de datos (digitales o analógicos), alimentación de dispositivos, como para el control de la propia placa [15]. En la Figura 8 se puede observar con mayor detalle la distribución de pines de la versión 3 de esta placa, aunque, como se ha mencionado anteriormente, no difiere mucho de las otras versiones anteriores.

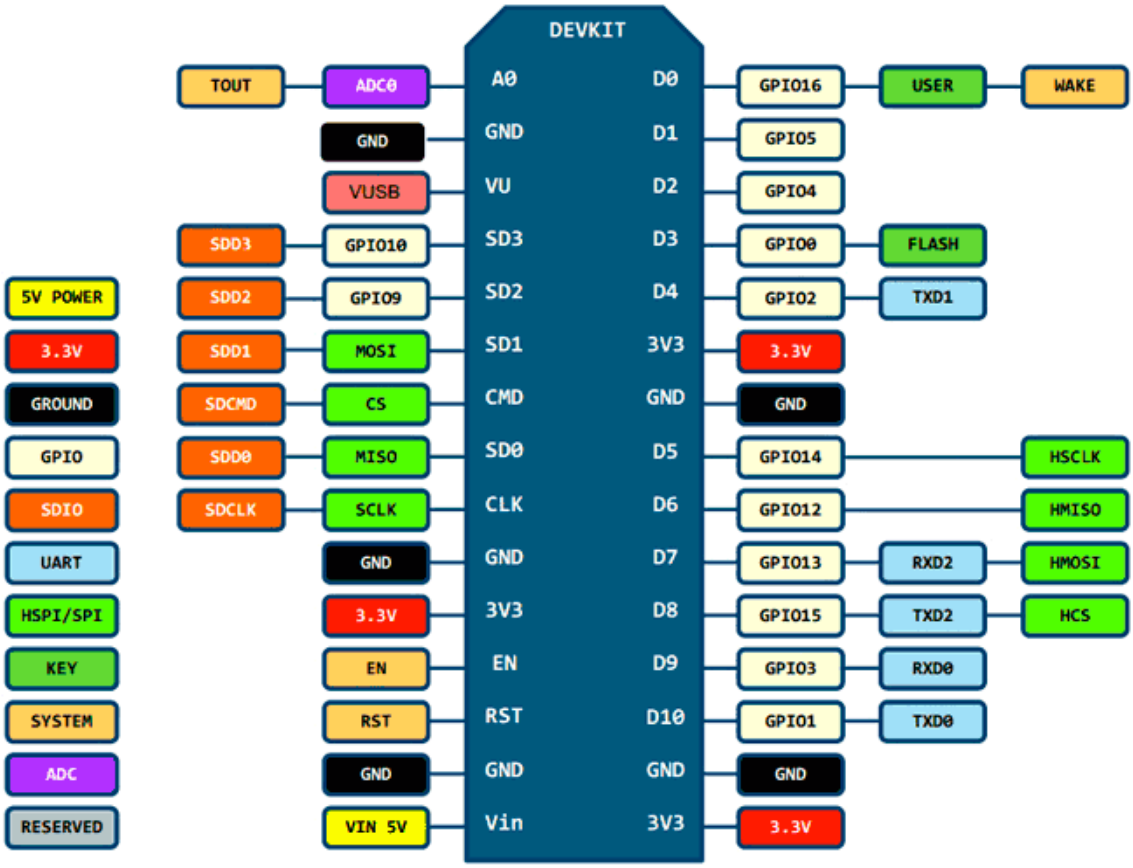


Figura 8. Pinout de NodeMCU v3.

En este proyecto se ha hecho uso de cinco placas de este tipo para diferentes propósitos:

- Control y envío de datos de sensor MQ135.
- Control y envío de datos de sensor MPU6050.
- Implementación de sniffer y envío de datos.

En cuanto a la programación de la placa, se pueden usar diferentes lenguajes como Arduino (basado en C/C++), Lua, Micropython… En este caso se ha optado por Arduino debido a la gran comunidad que lo respalda y a toda la documentación disponible. Para el desarrollo se ha empleado Arduino IDE, un entorno bastante sencillo diseñado específicamente para la programación con Arduino.

Para albergar el "nodo" central del sistema, la aplicación que recoge los datos y los almacena y envía, se va a hacer uso de un dispositivo con mayor capacidad computacional que los nombrados anteriormente. En este caso se ha decidido utilizar un mini ordenador que alcance estos requerimientos y, a su vez, sea asequible. De este tipo de dispositivos se encuentran varios en el mercado, como la famosa Raspberry, las placas de la compañía Libre Computer o el Odroid XU4, que tienen características muy similares. Por conveniencia, en este trabajo, se ha usado una Raspberry Pi 3.

Raspberry Pi 3 Model B

La Raspberry Pi es un ordenador de bajo coste (alrededor de 40 euros), de dimensiones reducidas, que puede conectarse a un monitor, teclado o ratón para su interacción, tal y como muestra la Figura 9. Según Raspberry Pi Fundación [16], "es capaz de hacer todo lo que se espera de un ordenador de sobremesa, desde navegar por Internet y reproducir vídeo de alta definición, hasta hacer hojas de cálculo, procesar textos y jugar".

En este caso se va a emplear el modelo B de la tercera generación de estos dispositivos, que incorpora conectividad WiFi y Bluetooth, además de diferentes puertos (USB, HDMI, etc.) y pines de entrada/salida a los que se pueden conectar componentes eléctricos, como leds.

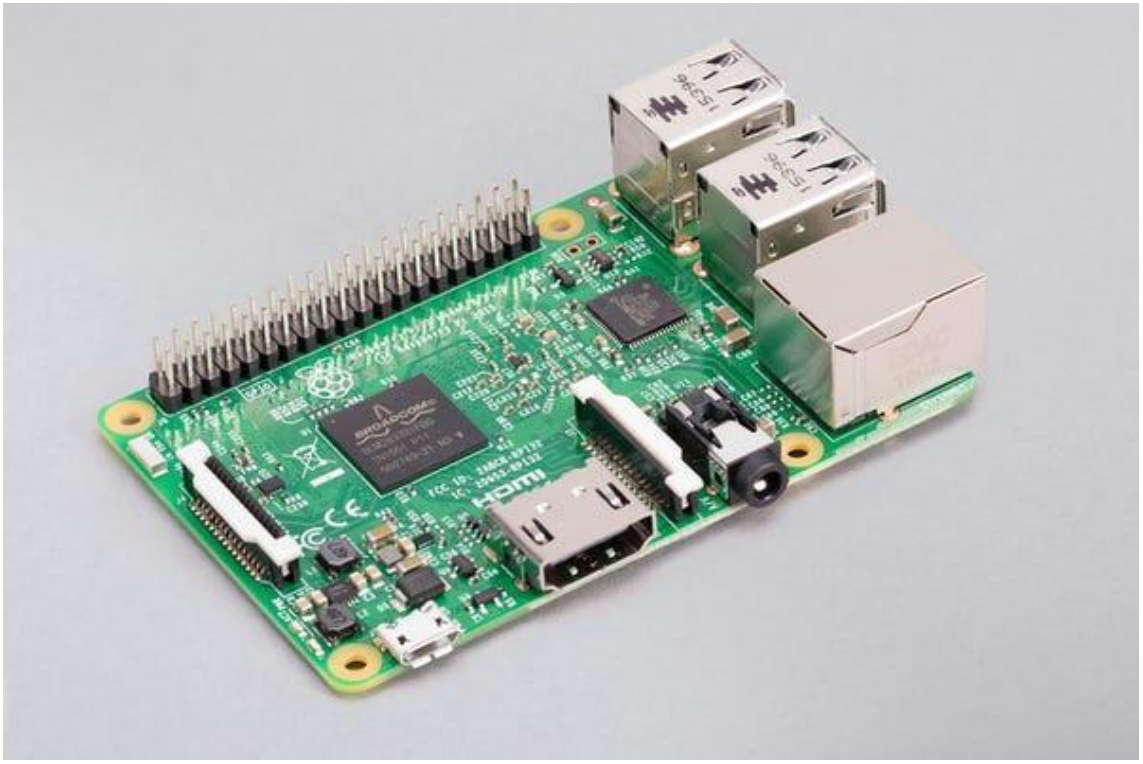


Figura 9. Raspberry Pi Model 3.

Como cualquier otro ordenador, dispone de sistema operativo, en este caso, el más común es Raspberrry Pi OS (anteriormente Raspbian) que está basado en Debian. En el anexo Configuración Raspberry. se describe su instalación mediante la utilidad software que ofrece la compañía.

Python y PyCharm

Para el desarrollo de la aplicación que recoge datos en la Raspberry se va a hacer uso del lenguaje Python en la versión 3 debido a las ventajas que tiene este lenguaje, entre ellas, la facilidad que supone, además de que viene incorporado en Raspberry Pi OS de forma nativa.

Como entorno de desarrollo para Python, Raspberry incluye Thonny, sin embargo, el desarrollo de la aplicación se va a realizar en el ordenador para mayor comodidad, por lo que finalmente, se usa PyCharm Community Edition 2021.2.2 de la compañía JetBrains. Esta herramienta es gratuita y ofrece muchas comodidades a la hora de escribir código, además de tener una

interfaz sencilla y bastante usable, lo que facilita mucho todas las tareas realizadas.

MongoDB

Como base de datos se ha empleado MongoDB, que es de tipo NoSQL (Not only SQL) y es una de las alternativas más utilizadas en la actualidad debido a las ventajas que supone, entre ellas [17]:

- Ofrece gran escalabilidad y flexibilidad, debido a que es una base de datos distribuida.
- Utiliza un modelo de documentos similares a JSON sin una estructura rígida, por lo que los campos pueden variar de unos documentos a otros.
- Es de uso gratuito.

Para hacer uso de MongoDB desde la aplicación Python se ha empleado la biblioteca Pymongo, que es la que se recomienda por parte de la documentación oficial. En el anexo Configuración Raspberry. se indica la sentencia para descargarlo e instalarlo.

Otro elemento a tener en cuenta es la plataforma IoT. Una plataforma de IoT permite la gestión de dispositivos conectados, la visualización, la gestión, el procesamiento y el análisis de datos, el desarrollo de aplicaciones, la seguridad, el control de acceso, la supervisión y el procesamiento de eventos. Hay numerosas alternativas en el mercado, desde las propietarias (Google Cloud Platform o IBM Watson IoT), hasta de código abierto, que son las que se han tenido en cuenta para la elección. Entre estas está la conocida Home Assistant, una plataforma de automatización del hogar que da prioridad al control local y a la privacidad, ideal para funcionar en una Raspberry Pi o en un servidor local. Además, destaca OpenHAB, con unas características muy similares con la anterior. Para este caso, se ha optado por usar ThingsBoard,

ya que tiene unas características similares y una gran comunidad que colabora y construye, además de su sencillez y usabilidad.

ThingsBoard y ThingsBoard Gateway

Como plataforma IoT se ha elegido ThingsBoard, que es de código abierto y permite la recopilación, el procesamiento, la visualización de datos y la gestión de dispositivos. Además, permite un rápido desarrollo, gestión y escalado de proyectos IoT. Esta plataforma ofrece una versión Cloud de pago, una versión Proffesional que incluye funcionalidad más avanzada y una versión Community Edition, que es la que se va a utilizar en este proyecto. Esta versión necesita de instalación, y , según la documentación oficial, es compatible con la mayoría de arquitecturas, desde Raspberry PI, hasta Google Cloud Platform [18], [19].

Algunas de las características de esta plataforma son:

- Escalabilidad horizontal.
- Es personalizable, ya que permite añadir nuevas funcionalidades de forma sencilla mediante widgets personalizables y nodos del motor de reglas.
- Comunicación vía MQTT, HTTP, LrM2M…

Con ThingsBoard se pueden abordar diferentes necesidades, incluso requisitos a nivel empresarial, entre las opciones de las que se dispone:

- Crear y manejar dispositivos, activos y clientes, y las relaciones que existan entre ellos.
- Recoger y visualizar datos de dispositivos y activos.
- Analizar los datos entrantes y activar las alarmas con el procesamiento de eventos complejos.
- Controlar los dispositivos mediante llamadas a procedimientos remotos (RPC).

- Construir flujos de trabajo basados en diversos eventos, como los relativos al ciclo de vida del dispositivo.
- Crear dashboards dinámicos y con capacidad de respuesta y que muestren a sus clientes los últimos datos recibidos de los dispositivos o de los activos.
- Manejar las funciones específicas para cada caso de uso mediante cadenas de reglas personalizables.
- Enviar los datos del dispositivo a otros sistemas.

Además de ThingsBoard, se ha hecho uso del módulo ThingsBoard IoT Gateway, que permite integrar dispositivos que están conectados a sistemas heredados y de terceros con la plataforma ThingsBoard IoT.

Por ejemplo, se pueden extraer datos de dispositivos que están conectados a brokers MQTT externos, servidores OPC-UA, Sigfox Backend, esclavos Modbus o nodos CAN. De esta forma, se podría tener un broker en una ubicación distinta a la máquina en la que se encuentra ThingsBoard, lo que ofrece una mayor flexibilidad y modularidad al sistema.

Se pueden visualizar algunos de los conectores disponibles de ThingsBoard Gateway en la imagen inferior.

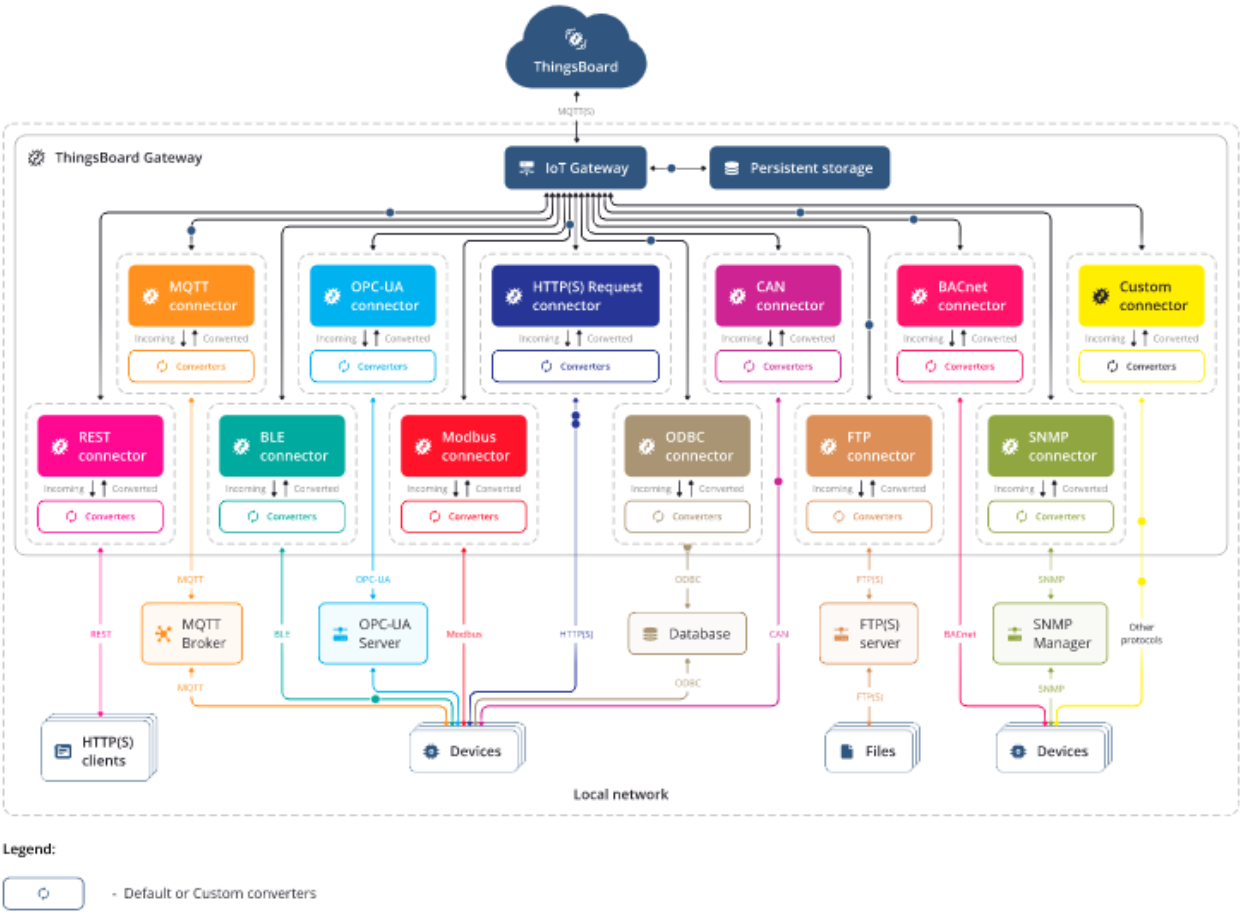


Figura 10. Estructura de ThingsBoard IoT Gateway.

Edge computing

Es un paradigma de computación que pretende acercar el procesamiento de los datos a la localización donde las cosas y las personas producen o consumen esa información, como dispositivos de IoT o servidores periféricos locales.

Este paradigma puede reducir los costes de la red, evitar las limitaciones de ancho de banda, reducir los retrasos en la transmisión, limitar los fallos del servicio y proporcionar un mejor control sobre el movimiento de datos sensibles. Los tiempos de carga se reducen y se añade la capacidad de realizar análisis y agregación de big data en el mismo lugar, que es lo que permite tomar decisiones casi en tiempo real.

2.2. Iteraciones.

El desarrollo del proyecto ha ido marcado por una serie de hitos correspondientes a cada uno de los componentes del trabajo en sí. Estos hitos van a definir los incrementos y las iteraciones que componen la metodología que se va a emplear.

2.2.1. Primera iteración. Primer prototipo básico

La primera iteración tiene como objetivo principal la familiarización con los sensores de gas MQ y la construcción de una aplicación de escritorio capaz de comunicarse con la placa y almacenar los datos en el mismo ordenador en el que se ejecuta. En este primer punto aún no se dispone de la mayor parte de elementos que componen este trabajo, únicamente se dispone de una placa NodeMCU con un sensor MQ2, muy parecido al sensor MQ135 que será el protagonista de este proyecto. Los motivos por los que se opta por este tipo de sensores son: son de bajo coste, funcionan de forma aceptable y la disponibilidad de estos a la hora de realizar el trabajo.

Por estos motivos, en este primer nivel se pretende conseguir el desarrollo de la aplicación básica (que es independiente del tipo de sensor que se utilice posteriormente) y el desarrollo del script que va a permitir la recogida de datos del sensor de gas de tipo MQ. En este primer paso también se establece la comunicación entre ambos componentes del sistema.

Análisis de requisitos.

En este caso, se dividen los requisitos en función de cada componente involucrado: aplicación de escritorio y dispositivo de medición.

- **Aplicación de escritorio**

  Los requisitos que se encuentran son:

  - Debe poder almacenar datos de forma local con el formato JSON específico que se muestra a continuación.

```
{
    "metadata": {
        "timestamp_end": "YYYY-MM-DD hh:mm:ss",
        "timestamp_start": "YYYY-MM-DD hh:mm:ss",
        "type": -
    },
    "samples": [
        {
            "property": -,
            "timestamp": "YYYY-MM-DD hh:mm:ss",
            "value": -
        },
        {
            "property": -,
            "timestamp": "YYYY-MM-DD hh:mm:ss",
            "value": -
        },
        ...
    ]
}
```

Figura 11. Estructura fichero JSON.

- Debe comunicarse de forma inalámbrica con el dispositivo de medición.
- El sistema debe permitir que el usuario ajuste el tiempo de muestreo y la frecuencia de almacenamiento de datos.
- Debe ser usable.

- **Dispositivo de medición**

  Los requisitos que se encuentran son:

  - Debe medir los datos de un gas específico en ppm (partes por millón) y de forma genérica en formato raw.
  - Debe comunicarse con la aplicación de forma inalámbrica.
  - Debe poder simular los diferentes estados de la puerta y ventana (abierto/cerrado). Estos estados se codifican según:
    - Si la puerta y la ventana están cerradas, “type A”.
    - Si la puerta está cerrada pero la ventada abierta, “type B”.
    - Si la puerta está abierta pero la ventana cerrada, “type C”.
    - Si la puerta y la ventana están abiertas, “type D”.

Diseño.

En el siguiente esquema de comunicación se representan los elementos principales que existen en la primera iteración.

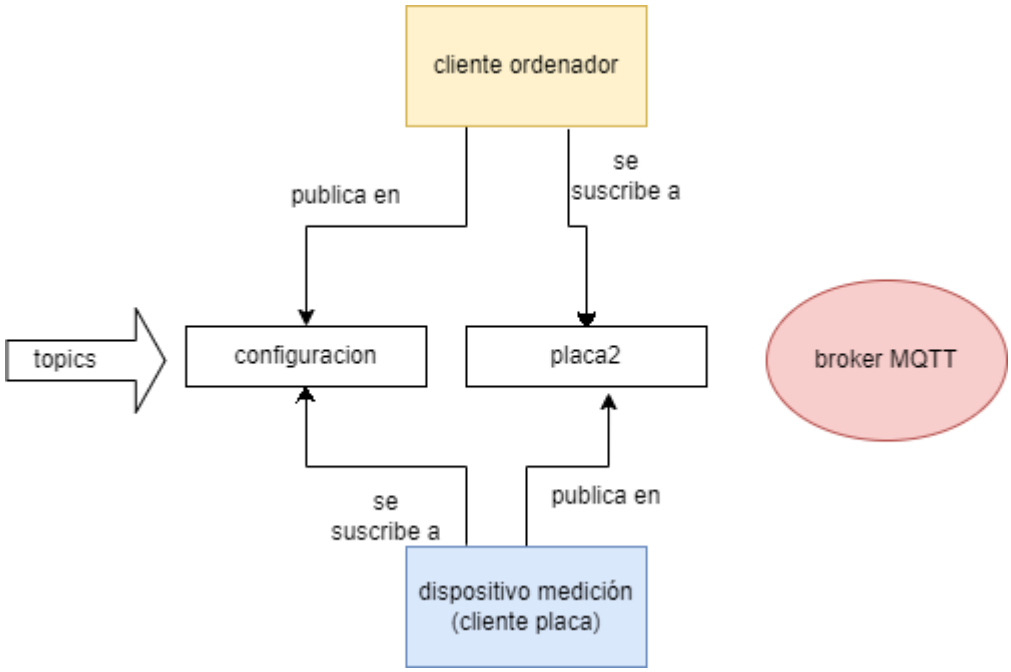


Figura 12. Comunicación MQTT entre los clientes.

- **Aplicación de escritorio:**

  Para el diseño arquitectónico de la aplicación se ha usado el paradigma de programación orientada a objetos. De esta forma, para implementar el comportamiento esperado de esta aplicación se va a hacer uso de:

  - El fichero GUI, que contiene la clase Window, que va a recoger el comportamiento de la interfaz gráfica con la que interactúa el usuario para establecer los tiempos en los que se va a almacenar el conjunto de muestras y la frecuencia con la que se van a tomar. Esta clase hace uso de la librería PyQt5 para crear y gestionar los elementos de la interfaz.
  - El fichero samples, compuesto por las clases Sample, Metadata y File:
    - La clase Sample, que contiene los atributos que representan a una muestra: timestamp, propiedad y valor.
    - La clase Metadata, que contiene los atributos que engloban al conjunto de todas las muestras (recogidas bajo unas características comunes): timestamp de inicio y final del conjunto

de muestras, el atributo type que hace referencia al estado de puerta y ventana y el id del sensor.

- La clase File, que hace referencia al conjunto de muestras y al metadata que las “recoge”. Esta clase se encarga de “pasar” los datos de los objetos a formato JSON para que sean escritos a posteriori.

- El fichero utils, que contiene la clase Tiempo, de tipo Enum, para poder cuantificar los intervalos de almacenamiento y muestreo. Permite pasar las horas/minutos/segundos/milisegundos a segundos y milisegundos dependiendo de la función que tenga cada valor.
- El fichero client, que va a gestionar toda la funcionalidad principal de la aplicación.

- **Dispositivo de medición:**

  Para implementar el comportamiento del dispositivo de medición se va a hacer uso del modelo de programación *semi estructurada* que siguen las placas Arduino. Normalmente, estos dispositivos suelen tienen una funcionalidad limitada a varias acciones (tomar mediciones, enviarlas, realizar acciones sobre el entorno…). Por eso se implementan en lo que se denomina sketch, pequeños scripts en los que se codifica todo el comportamiento. Por ese motivo, no se ha creado ninguna clase para modelar este comportamiento tan básico y se ha utilizado programación estructurada.

  En los sketches de Arduino, el código se divide en dos métodos principales: setup y loop. El primero de ellos hace referencia a las instrucciones que deben realizarse al inicio de la ejecución del programa una única vez, mientras que el segundo es un método que se repite infinitamente tras la ejecución del setup.

  La única decisión de diseño en este punto ha sido incluir botones/pulsadores en el circuito para poder introducir información relacionada con el estado de la puerta y ventana.

Implementación.

- **Aplicación de escritorio:**

  La aplicación se encarga de recibir los datos, darles el formato adecuado y guardarlos en archivos cada cierto tiempo. Para ello, almacena las muestras en diccionarios y para cada uno de los dispositivos que pueden enviar datos, mantiene un temporizador para controlar el tiempo en el que se van a almacenar el conjunto de muestras.

  Se inicializa el cliente principal MQTT con los métodos `onConnect, onMessage, onPublish y connect(brokerIP, 1883, 60).`

  El cliente se suscribe a los topics donde van a publicar los datos los dispositivos con el método `susbcribe(topic).`

  Mientras tanto, el hilo principal se encarga de mostrar la interfaz gráfica para el usuario, en la que los botones permiten cambiar los tiempos de guardado y medición de muestras. En el caso de las mediciones, cuando se decida cambiar de intervalos temporales, se creará un nuevo cliente mqtt solo para publicar los datos.

  Cuando llega un mensaje mediante MQTT, el callback on_message lo recibe y almacena las muestras. Mira el tipo (según los botones) para saber si hay que escribir las muestras en el fichero. Si no, se crea un temporizador (con la librería Threading) para cada dispositivo y se le asigna como argumento la función que será la encargada de escribir la información en los ficheros una vez pase el tiempo establecido, de forma que los datos almacenados “en caché” se escriben en un fichero con el formato indicado anteriormente.

- **Dispositivo de medición**

  Para el dispositivo de medición hay que distinguir entre el montaje del circuito eléctrico y el código con el que se va a programar el microcontrolador (NodeMCU).

  En primer lugar, se presenta el montaje del circuito, en el que se usan tres botones para representar los diferentes estados de la puerta y ventana

codificados como type A, type B, type C o type D. Para conocer qué estado es el que se encuentra activo, se añaden tres luces led que se iluminan según qué botón se haya pulsado. Por otra parte, se incluye el sensor MQ2, aunque, en este caso, no está alimentado correctamente debido a la versión de la placa NodeMCU de la que se dispone (se detallará en la tercera iteración).

Además de los componentes principales hay que incluir resistencias para evitar cortocircuitos (en el caso de los botones) o para alargar la vida de las luces led.

Para conectar los botones se utiliza el mecanismo pull down, en el que cuando se pulsa el botón, la salida digital del circuito es el valor alto (HIGH), mientras que el resto del tiempo la salida digital se mantiene a nivel bajo (LOW). Por otro lado, cada una de las salidas digitales de los tres botones se va a conectar a uno de los pines digitales de la placa (D1, D2 y D3).

Para conectar el sensor, hay que conectar la salida analógica de este (AO) al único pin analógico de la placa (A0), la alimentación (VCC) al pin de la placa que provee 3.3V y la tierra al pin GND de la placa.

Todo este montaje viene descrito de forma gráfica en la Figura 13.

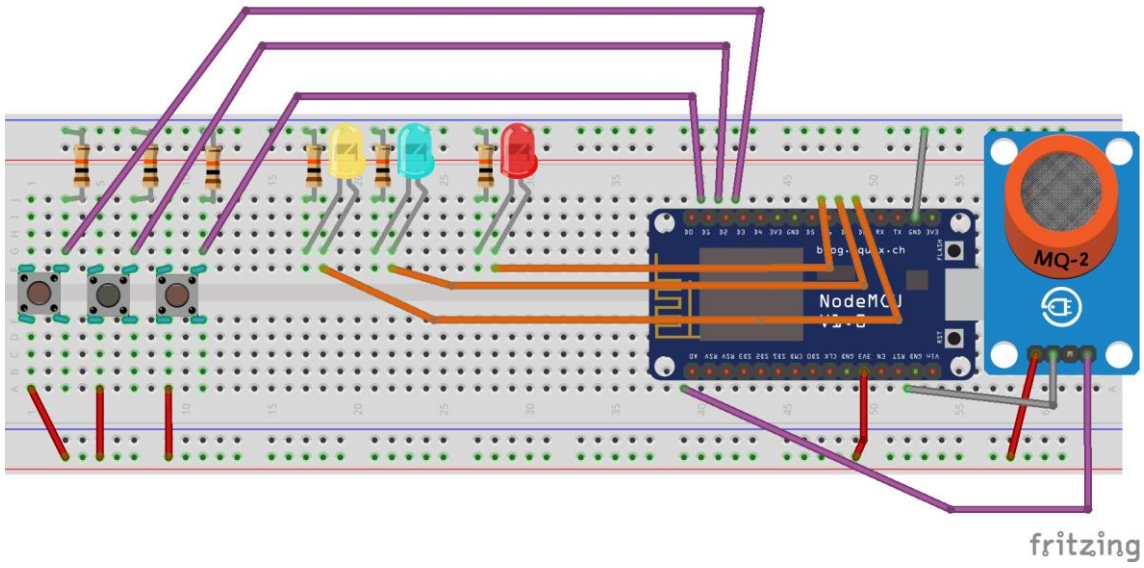


Figura 13. Esquemático del circuito.

Una vez realizado el montaje, aunque en una breadboard más pequeña, quedaría como se muestra en la Figura 14.

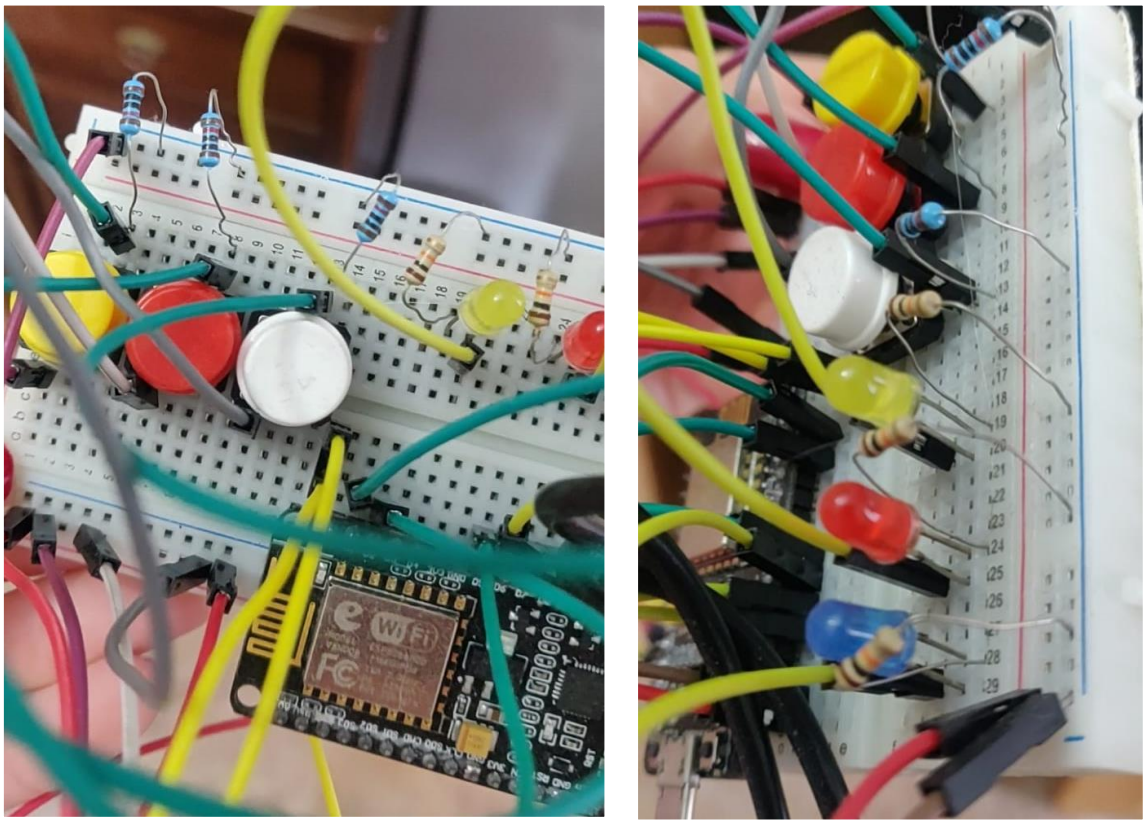


Figura 14. Imagen del montaje eléctrico.

En cuanto al código que se va a grabar en la placa, destaca:

El uso de la biblioteca PubSubClient como cliente del protocolo MQTT adaptado a las características de la placa. Esta librería permite el envío y recepción de mensajes mediante MQTT de forma sencilla:

- Especificar los parámetros de conexión, con los parámetros en el constructor adecuados: `client(ipBroker, 1883, callback, wifiClient)`.

  Cabe destacar que 1883 es el puerto que suele usar MQTT por defecto, callback es la función a la que llegan los mensajes de los topics a los que el cliente se suscribe y wifiClient hace referencia a la forma en la que se establece la conexión, en este caso, mediante WiFi.

- Conectarse al broker, mediante la orden `client.connect(nombreCliente)`
- Suscribirse a los topics, mediante la orden `client.subscribe(topic)`
- Publicar mensajes, mediante la orden `client.publish(topic, mensaje)`

Para la recogida de datos del sensor, se va a usar la librería MQUnifiedSensor, que es capaz de interpretar los datos recogidos por este y discernir entre diferentes tipos de gases.

En el método setup se inicializan las variables que indican el estado de los botones por defecto (nivel bajo), se indica si los pines de los botones y los leds son de entrada o salida, se configuran los parámetros de la conexión Wifi, se inicializa el objeto que recoge la configuración del sensor y se establece la comunicación mediante MQTT.
En el método loop, que se repite infinitamente, en primer lugar, se comprueba la conexión con el broker MQTT para volver a establecerla en caso necesario. A continuación, se comprueba si se ha pulsado algún botón para actualizar las variables que indican el estado. Posteriormente se realiza la medición con el método getPPM que provee la clase MQUnifiedSensor y se publican tanto estos datos como la información raw medida por el sensor y el estado marcado por los botones en el topic *placa2.* Cabe mencionar que las mediciones y la publicación de los datos se realizan exclusivamente cada cierto tiempo, al que se denomina frecuencia de muestreo. Este tiempo puede ser ajustado gracias a que el dispositivo se suscribe al topic *configuración*, donde recibe el número de milisegundos enviados por el cliente de escritorio.

Pruebas realizadas.

Las pruebas realizadas se basan en la ejecución de la aplicación de escritorio y la puesta en marcha del dispositivo. Se prueba a ajustar los diferentes tiempos

y se comprueba que los ficheros son correctos. Además, se comprueba el correcto funcionamiento del mecanismo de botones.

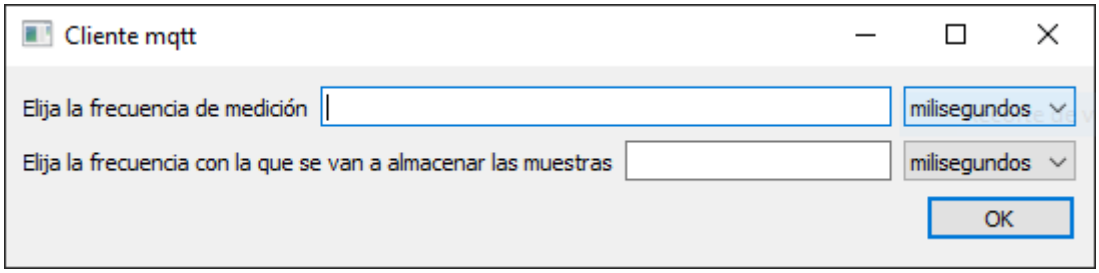


Figura 15. Interfaz para el ajuste de tiempos de guardado y recogida de muestras.

Como resultado, se obtienen ficheros como el siguiente:

Figura 16. Ejemplo de fichero resultante de las pruebas realizadas.

En esta primera iteración se pueden obviar los resultados cuantificables de las medidas de gas obtenidos con este sensor MQ2 debido a que no mide concentraciones de CO_2. Lo que se pretende es que las comunicaciones entre la placa y la aplicación sean correctas y que la salida tenga el formato adecuado.

2.2.2. Segunda iteración. Mecanismo de apertura.

En este punto se pretende reutilizar la aplicación desarrollada anteriormente y añadir la utilidad para detectar la apertura o cierre de la ventana o puerta.

Búsqueda de información y documentación.

En este punto hay que considerar la búsqueda de información y pruebas con los diferentes sensores considerados para elegir el que sea más adecuado.

En primer lugar, se considera el uso de un acelerómetro para lograr conocer el estado de la puerta y ventana. Concretamente se usa el acelerómetro MMA 8452, pequeño y de bajo consumo, ideal para este caso de uso.

Para conocer el funcionamiento de este acelerómetro, se descargó código de ejemplo de uno de los fabricantes más conocidos, Adafruit. En el ejemplo, se muestra cómo configurar y conectar correctamente el acelerómetro mediante I^2C, y haciendo uso de las librerías Adafruit_MMA8451 y Adafruit_Sensor permite conocer la aceleración en cada eje y la orientación del dispositivo (si mira hacia arriba o hacia abajo o si está en modo landscape o portrait). Al ejecutar este ejemplo, la salida muestra los datos raw de aceleración y, expresados en m/s^2, de un modo más entendible. Además, también se obtiene la información relativa a la orientación, tal y como se muestra en la Figura 17. Como se puede observar, el acelerómetro marca de forma evidente el eje sobre el que influye la gravedad, ya que los valores de aceleración en este van a ser siempre en torno a los 9.81 m/s^2. En cuanto a los demás ejes, al simular el movimiento de apertura de la puerta, muestran valores de aceleración, aunque varían en las diferentes ejecuciones haciendo que representar el movimiento no sea una tarea tan trivial.

Tras diversas pruebas, se llega a la conclusión de que el acelerómetro no es idóneo para el tipo de movimiento circular que se realiza al abrir la puerta y ventana. En caso de que el movimiento implicara un cambio en la orientación del dispositivo (cambio en los ejes sobre el que afecta la gravedad), sí podría ser suficiente, pero en este caso donde la velocidad de apertura puede ser constante, el resultado no es adecuado.

```
08:23:26.406 -> X:      4032    Y:      360     Z:      -140
08:23:26.406 -> X:      9.65    Y:      0.89    Z:      -0.36   m/s^2
08:23:26.453 -> Landscape Right Front
08:23:26.499 ->
08:23:28.415 -> X:      4012    Y:      384     Z:      -172
08:23:28.415 -> X:      9.62    Y:      0.88    Z:      -0.37   m/s^2
08:23:28.462 -> Landscape Right Front
08:23:28.508 ->
08:23:30.424 -> X:      4032    Y:      368     Z:      -156
08:23:30.424 -> X:      9.62    Y:      0.84    Z:      -0.38   m/s^2
08:23:30.471 -> Landscape Right Front
08:23:30.471 ->
08:23:32.432 -> X:      4024    Y:      360     Z:      -164
08:23:32.432 -> X:      9.62    Y:      0.85    Z:      -0.40   m/s^2
08:23:32.479 -> Landscape Right Front
08:23:32.479 ->
08:23:34.399 -> X:      4032    Y:      368     Z:      -152
08:23:34.446 -> X:      9.62    Y:      0.89    Z:      -0.38   m/s^2
08:23:34.492 -> Landscape Right Front
08:23:34.492 ->
08:23:36.410 -> X:      4028    Y:      372     Z:      -176
08:23:36.457 -> X:      9.62    Y:      0.87    Z:      -0.39   m/s^2
08:23:36.504 -> Landscape Right Front
08:23:36.504 ->
08:23:38.416 -> X:      4024    Y:      364     Z:      -172
08:23:38.463 -> X:      9.62    Y:      0.86    Z:      -0.42   m/s^2
08:23:38.463 -> Landscape Right Front
```

Figura 17. Salida de la prueba con el acelerómetro.

A continuación, se cambió el sensor por el módulo MPU 6050, que incorpora un giroscopio, además de un acelerómetro. Gracias a este, se puede obtener la velocidad angular, y de ahí descubrir el ángulo de apertura. La velocidad angular es la tasa de cambio del desplazamiento angular por unidad de tiempo. Por tanto, para conocer el ángulo de apertura, hay que utilizar la ecuación que relaciona el ángulo con la velocidad angular:

$$\omega = \frac{d\theta}{dt}$$

donde:

$\omega\ es\ velocidad\ angular, d\theta\ es\ la\ diferencia\ angular\ respecto\ instante\ de\ tiempo\ (dt).$

De esta forma, integrando con respecto al tiempo, se obtiene que:

$$\theta = \omega * \Delta t$$

donde:

$\theta, es\ el\ ángulo\ desplazado\ por\ la\ velocidad\ angular\ \omega\ en\ un\ intervalo\ de\ tiempo\ \Delta t.$

Con lo que, si se conoce la velocidad angular en un momento de tiempo y el tiempo que ha transcurrido desde la última medición, se podrá obtener el ángulo total de rotación mediante la acumulación de los ángulos desplazados en los instantes actual y anteriores.

El problema con este cálculo es que a medio y largo plazo sufre deriva, lo que hace que haya diferencias entre el ángulo que se obtiene y el ángulo real. Por ejemplo, al realizar las pruebas, el ángulo cuando la ventana está cerrada es 0, mientras que, si la se abre un poco y se vuelve a cerrar, el valor obtenido difiere unos cuantos grados cuando se sabe que debería ser 0. Este error se va acumulando con el tiempo y el movimiento, lo que hace que cada vez sean menos precisas las mediciones calculadas con la fórmula anterior. Para intentar paliar este efecto, se ha ideado un método para calcular cuándo la puerta está totalmente abierta y cuándo está totalmente cerrada. Para ello, se obtienen los últimos valores que ha medido el giroscopio y se calcula un error absoluto entre estas muestras. Si este valor es lo suficientemente bajo, se puede afirmar que la puerta/ventana se encuentran quietas (estáticas) y en ese caso, comprobar si está abierta/cerrada dependiendo del ángulo total calculado y un umbral que determina este estado (por encima de X grados se considera abierta y viceversa).

Análisis de requisitos.

Los requisitos que se encuentran son:

- Debe poder simular los diferentes estados de la puerta y ventana (abierto/cerrado).
- Debe ajustarse al tipo de puerta y ventana de la estancia. En este caso, tanto la ventana como la puerta se abren de la misma forma, girando sobre su eje vertical.
- Debe poder enviar los resultados mediante MQTT en el formato correcto.
- El tiempo de detección debe ser razonable, para que, si se realizan cambios puedan detectarse.

Diseño.

En este punto se produce un cambio en cuanto a los elementos utilizados, y se separa cada funcionalidad en un elemento diferente. En el diagrama de secuencia inferior se pueden observar dichos elementos, su comportamiento y la forma de comunicación que va a haber entre ellos. Los detalles e implementación se mostrarán en las iteraciones posteriores.

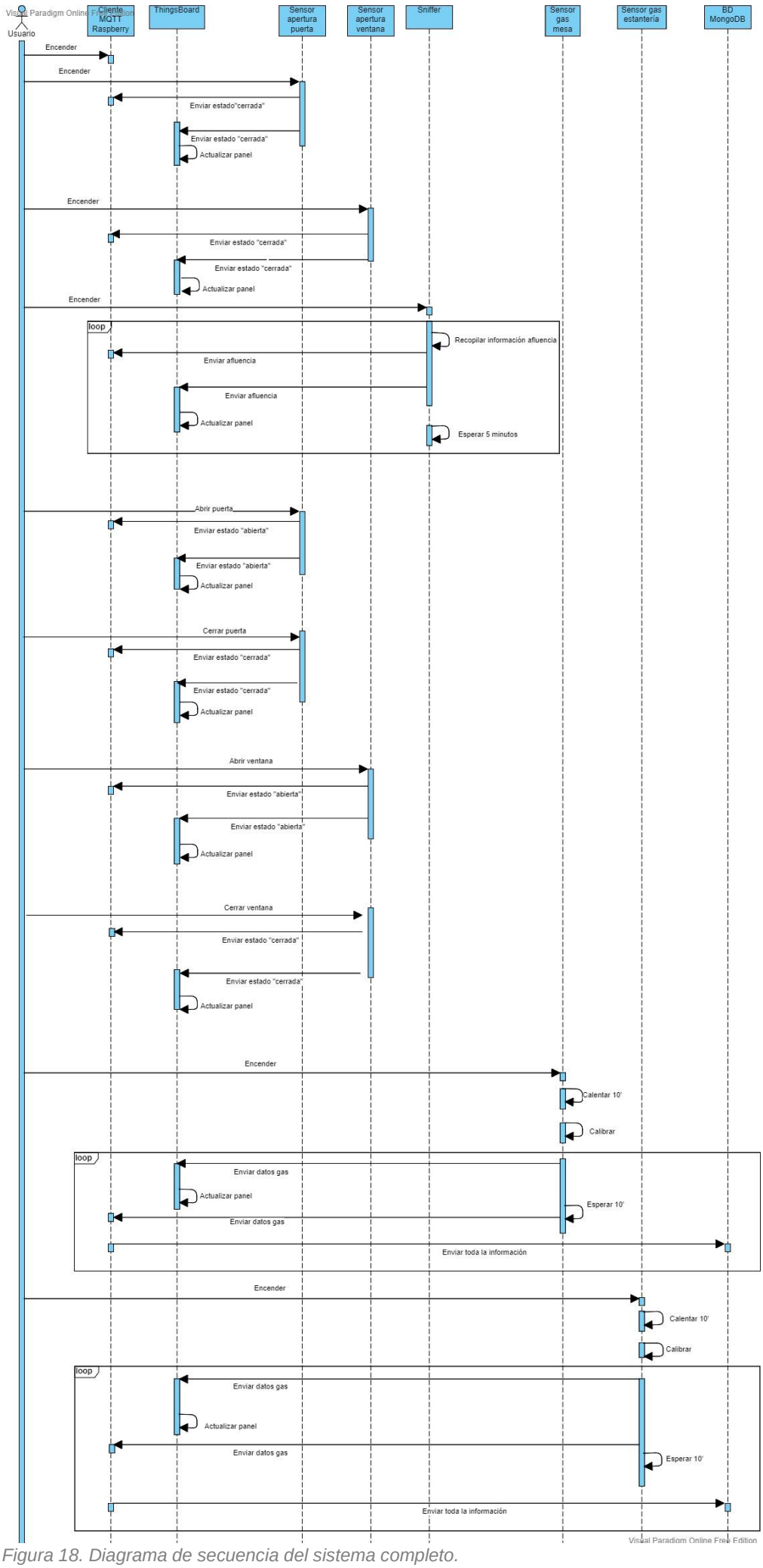


Figura 18. Diagrama de secuencia del sistema completo.

- **Dispositivo de medición:**

En este punto, como el sistema se ha dividido en varios dispositivos focalizados cada uno en una tarea, el modo de envío de mensajes mediante MQTT va a realizarse haciendo uso del formato JSON para conseguir una mayor estandarización. El dispositivo medirá los datos y los enviará al broker para que el resto del sistema los utilice. Para ello, cada uno de los dispositivos tendrá asociado un topic en el que publicará y al que se suscribirá, que deberá ser compartido por la aplicación sumidero para la comunicación.

En este proyecto la información que se precisa es conocer el estado de la puerta y ventana, no los ángulos de apertura. Sin embargo, con el giroscopio se obtienen las medidas como ángulos, por lo que habría que fijar una cifra (en grados) sobre la que se decida el estado, de tal forma que, si el ángulo medido es menor que esa cifra, se considere cerrada, y si es mayor, abierta.

Para solventar el fenómeno de la deriva de forma sencilla y sin un alto coste computacional, se ha decidido hacer “reinicios” en los cálculos, de forma que cuando se cierra la puerta o ventana, se ajusta el ángulo a 0 (ya que es el punto de partida), lo que hace que se detenga la acumulación de deriva y los cálculos de ángulos vuelvan a comenzar desde el punto de partida.

Implementación.

- **Dispositivo de medición:**

En cuanto al código del script que se va a grabar en la placa:

En el primer método que se va a ejecutar (`setup`) se inicializan las variables y el objeto que representa al giroscopio (biblioteca MPU6050), además de definir el estado inicial como “cerrada”.

Por otra parte, se tiene el método `updateGiro`, que se encargará de realizar operación acumulativa de ángulo con respecto al tiempo, obteniendo así el ángulo total de apertura.

Este dispositivo va a estar midiendo datos constantemente, no tiene ningún tiempo en pausa o “dormir”, ya que tiene que comprobar los ángulos continuamente porque es impredecible el momento en el que se va a producir un evento de abrir/cerrar. Por este motivo, en un pequeño instante puede recoger varias mediciones. Es por esto que hay que considerar que para que la puerta o ventana cambie de estado debe permanecer en este estado un cierto tiempo (que no sea un movimiento rápido y vuelva a la posición original).

En el método que se ejecuta infinitamente, `loop`, se llama a `updateGiro` y, con esa información, pasa a detectar si la puerta/ventana se encuentra estática para poder enviar la información correcta. Para ello, lo que se hace es almacenar en una cola los últimos diez ángulos medidos y calcular el error absoluto con respecto al último recogido para así conseguir cierta estabilidad. En el caso de que este error sea cercano a 0, se procede a comprobar si está por debajo o por encima del umbral fijado para así determinar su estado (abierta o cerrada) y “reiniciar el ángulo”, de forma que, si se considera abierta, el ángulo sería 90º y si está cerrada, 0º, para contrarrestar el efecto de la deriva. Una vez sabido ese estado, se procede a enviar la información al broker MQTT haciendo uso de la librería ArduinoJson, para que posteriormente se pueda extraer dicha información con mayor facilidad.

Para la parte del montaje electrónico:

En cuanto al montaje (Figura 19 y Figura 20), aunque el módulo dispone de ocho pines, en esta ocasión para esta aplicación únicamente habría que conectar cuatro de ellos:

- la alimentación (VCC), que se conecta con el pin 3.3V de la placa, ya que este sensor si trabaja con ese nivel de voltaje.
- tierra (GND), que se conecta con el pin GND de la placa.
- los correspondientes al protocolo de comunicación I2C (SCL y SDA), que se conectan con los pines D1 y D2 de la placa, respectivamente.

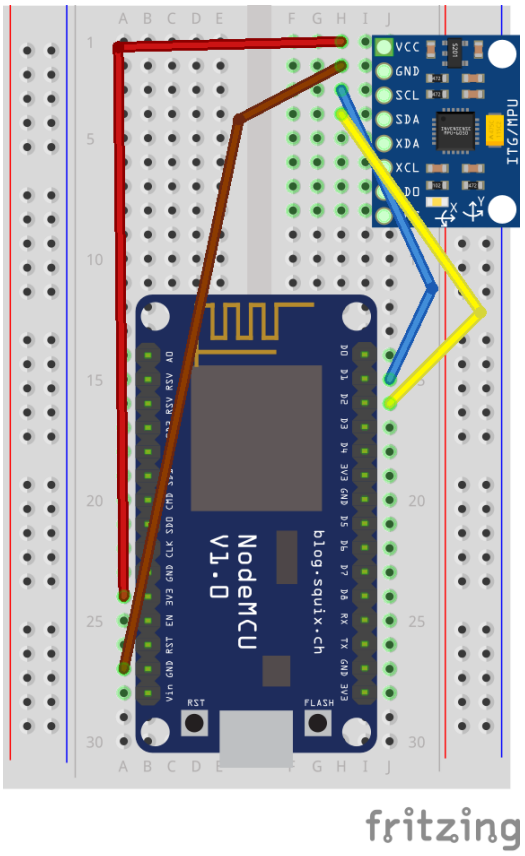


Figura 19. Esquemático del montaje con el módulo MPU 6050.

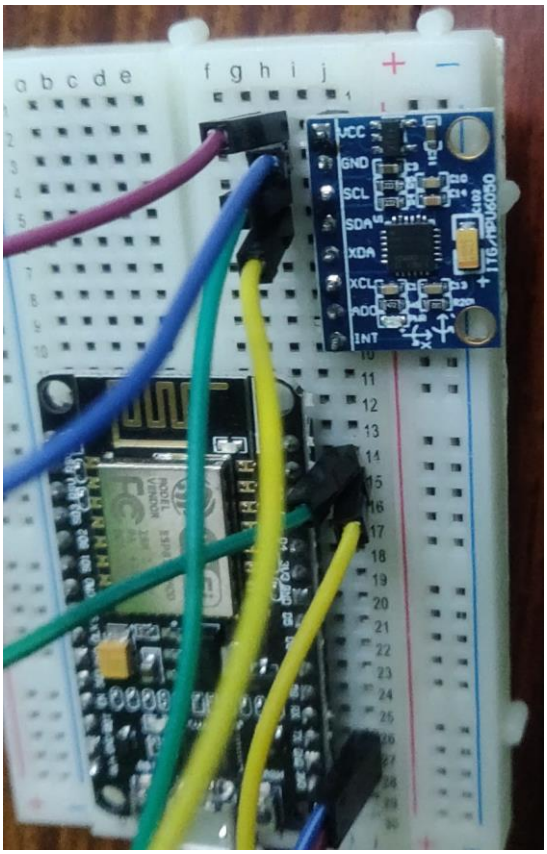


Figura 20. Montaje de placa con módulo MPU 6050.

Hay que tener en cuenta la orientación del dispositivo (hacia abajo), de forma que el eje Y quede alineado con la gravedad, que es sobre el cual se van a realizar las mediciones (Figura 21).

Figura 21.Colocación de la placa y la batería en la puerta y ventana.

Pruebas realizadas.

Las pruebas que se han realizado constan de diferentes intentos de apertura y cierre tanto de la puerta como de la ventana en diferentes velocidades, para comprobar que el estado que se registra es igual a la realidad y, por lo tanto, correcto. También se comprueba que los mensajes que se envían desde la placa sean correctos en cuanto a formato.

2.2.3. Tercera iteración. Detección de CO_2.

En este punto se pretende incluir el sensor adecuado para la medición de CO_2 en las placas con los sensores definitivos basándose en la primera iteración. En este caso se dispone de dos sensores ubicados en las esquinas opuestas de la habitación.

Análisis de requisitos.

Los requisitos que se encuentran son:

- Debe medir la concentración de CO_2 y expresarla tanto en partes por millón (ppm) como en formato raw.

- Debe poder enviar los datos mediante MQTT en un formato adecuado.
- Debe poder ajustar el tiempo de muestreo dinámicamente.

Diseño.

- **Dispositivo de medición:**

Una vez realizada la lectura de la hoja de características del sensor MQ135, se deduce que es necesario un pre calentamiento para la correcta medición de los datos, por ello hay que mantener el dispositivo en funcionamiento, pero solo para que la alimentación permita obtener cierta temperatura.

El diseño que se va a seguir es el que se mostró anteriormente. El dispositivo medirá los datos cada cierto tiempo (podrá configurarse) y los enviará al broker para que el resto del sistema los utilice. Para ello, cada uno de los dispositivos tendrá asociado un topic en el que publicará y al que se suscribirá, que deberá ser compartido por la aplicación sumidero para la comunicación.

Por otra parte, se debe poder configurar la frecuencia de medición del gas desde la aplicación sumidero del ordenador, acción similar a la primera iteración, en la que este dato se comunicaba mediante MQTT.

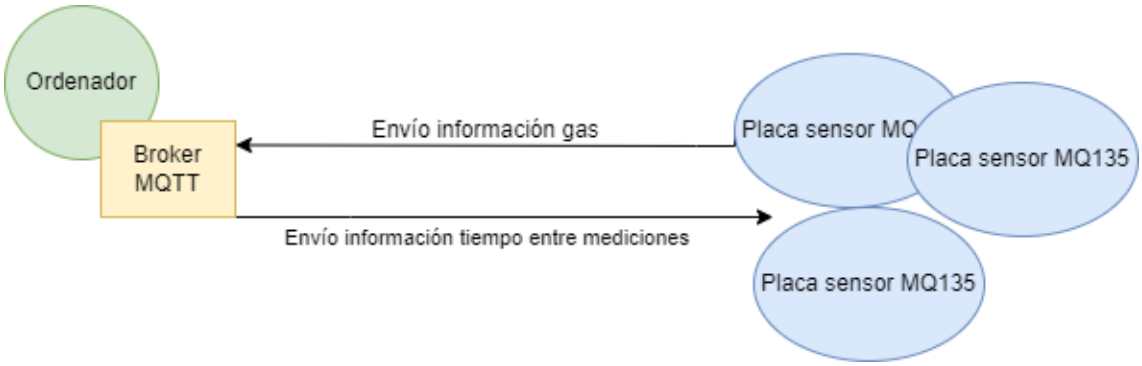


Figura 22. Esquema de comunicación sensores de gas y aplicación de escritorio.

Implementación.

- **Dispositivo de medición.**

Para el montaje de los componentes hardware es necesario tener en cuenta:

- Que el sensor de gas necesita un voltaje de 5V para funcionar, y su salida analógica también es de 5V.
- Que el pin analógico de la placa Node MCU admite un voltaje de entre 0V y 3.3V.

Por este motivo, hay que ajustar los valores de salida del sensor de gas para que sean adecuados en las lecturas analógicas. Para ello, es necesario incluir un divisor de voltaje mediante el uso de resistencias de 10 kΩ entre la salida y el pin A0.

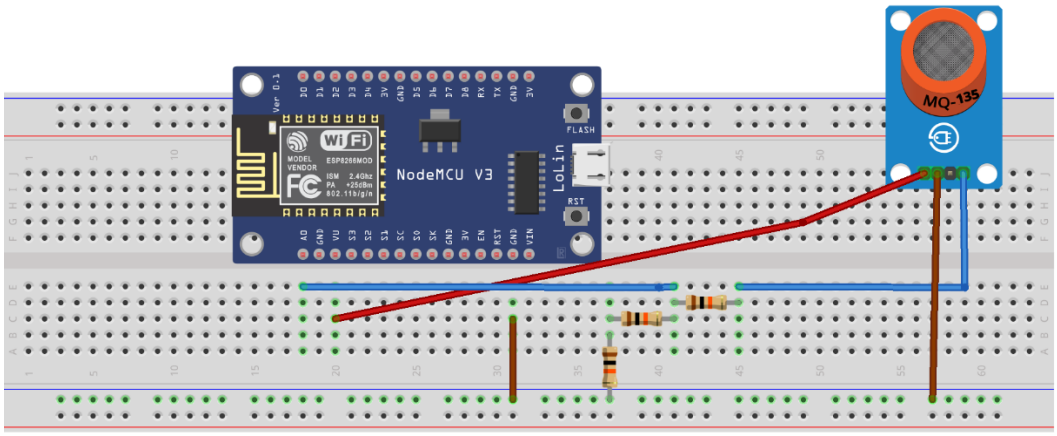


Figura 23. Esquemático del montaje del sensor MQ 135.

En la Figura 24 se puede apreciar el montaje de los componentes.

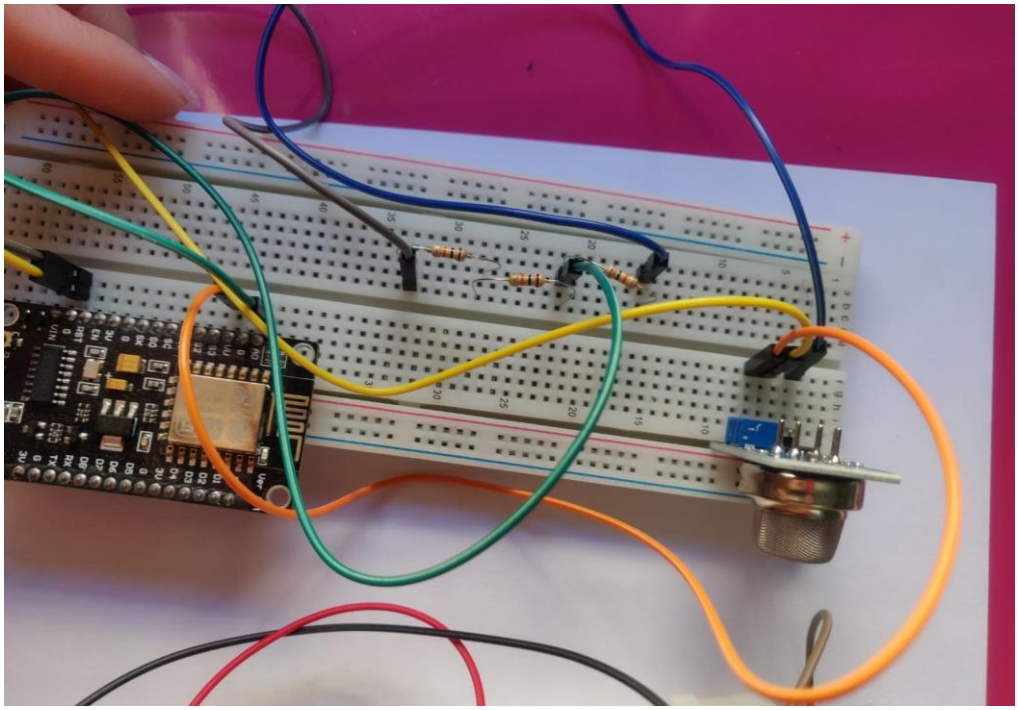


Figura 24. Montaje dispositivo de medición de gas con MQ135.

En cuanto al código, para que las medidas sean correctas, como se ha mencionado anteriormente, es necesario que el sensor esté caliente antes de comenzar a tomar en cuenta las mediciones. Para ello, en el código se ha incluido una parte en la que se emplean diez minutos para esta acción en la que al dispositivo se le suministra energía para adquirir la temperatura óptima de trabajo.

Además de este calentamiento, tanto las condiciones ambientales de temperatura y humedad van a afectar a las medidas de CO_2. Por este motivo, es necesario realizar una calibración del sensor con aire “limpio” antes del uso, que realiza la propia librería.

Para la implementación de esta funcionalidad, se ha hecho uso de la librería MQUnifiedSensor, cuyo uso está bastante extendido cuando se trata de los sensores de gas de la familia MQ. Su uso es muy sencillo, simplemente hay que configurar unos parámetros relacionados con el hardware utilizado (Figura 25).

```
#include <MQUnifiedsensor.h>

// DATOS PARA SENSOR MQ135
/************************Hardware Related Macros************************************/
#define Board ("ESP8266")
#define Pin (A0)

/***********************Software Related Macros************************************/
#define Type ("MQ-135")
#define Voltage_Resolution (3.3)
#define ADC_Bit_Resolution (10) // For ESP8266
#define RatioMQ135CleanAir (3.6)

MQUnifiedsensor MQ135(Board, Voltage_Resolution, ADC_Bit_Resolution, Pin, Type);
```

Figura 25. Parámetros a especificar en la biblioteca MQUnifiedSensor.

Además de estos parámetros, para obtener los valores específicos de CO_2, es necesario introducir los valores (llamados a y b) que se sustituirán en la ecuación de regresión para calcular las partes por millón de este gas. Para este caso, el valor de a es de 110.47 y el de b es de -2.862. Una vez configurados todos estos parámetros, se procede a la calibración del sensor para calcular los valores del "aire limpio" sobre los que partirán los valores que se calcularán dentro de la estancia. Se puede decir que se calcula el valor cero mínimos que corresponde al nivel de CO_2 que se encuentra en el exterior.

Después de la calibración, comenzaría a ejecutarse el código que se va a repetir indefinidamente mientras los componentes estén alimentados. Este código se limita a actualizar y leer los valores del sensor y enviarlos al servidor mediante MQTT. Previamente se comprueba la conexión con el servidor y se vuelve a conectar en caso de que esta haya finalizado. Los diferentes datos (id del sensor, ppm de CO_2 y datos raw) se envían con formato JSON, haciendo uso de la librería ArduinoJson.

Además, se configura el callback MQTT para que sea capaz de extraer los mensajes de configuración de tiempo de muestreo y actualizar dicho tiempo.

Pruebas realizadas.

Las pruebas realizadas consisten en la ejecución del código de las placas y la comprobación del correcto envío de los datos con el formato en el que se ha implementado. También se comprueba que la configuración del tiempo de

muestreo funciona, mediante la publicación de mensajes en el topic de configuración de cada uno de los dispositivos.

2.2.4. Cuarta iteración. Adaptar la aplicación de recogida de datos.

En este punto se pretende reutilizar la aplicación desarrollada anteriormente y adaptarla a los cambios que se han producido al separar las funcionalidades de medir el gas y controlar los estados de la puerta y la ventana. En este momento, los diferentes datos van a llegar en momentos distintos (o no) y se van a realizar cambios en cuanto al almacenamiento de estos.

Análisis de requisitos.

Los requisitos que se encuentran son:

- Debe poder cambiar el tiempo de muestreo de cada dispositivo de medición de gas. Para ello ha de utilizar un fichero de configuración.
- Debe almacenar toda la información recogida por los sensores en la base de datos MongoDB indicada para ello.

Diseño.

No se han modificado las clases que ya existían en la primera aplicación, simplemente se ha modificado el código para el almacenamiento de las muestras.

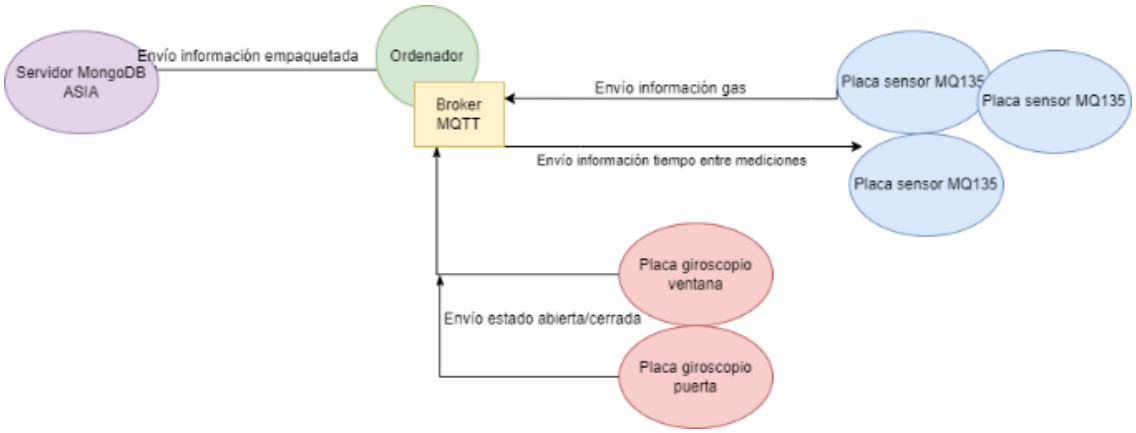


Figura 26. Estructura comunicación sensores-aplicación-servidor.

Implementación.

Para el almacenamiento de los datos, se pone a disposición acceso un servidor de base de datos MongoDB remoto del grupo ASIA, en el cual me crean un usuario administrador para gestionar las diferentes opciones. Posteriormente, se crea una base de datos que va a almacenar la información relativa a este trabajo, llamada tfm. Para la conexión a la base de datos es conveniente la creación de otro usuario cuyos privilegios sean limitados.

En la Figura 27 se muestra la creación del usuario clienteMQTT, con privilegios de lectura y escritura y acceso a la base de datos tfm, y que se va a utilizar en la aplicación de escritorio para la inserción de documentos.

Figura 27. Creación de usuario en MongoDB.

En el código de la aplicación, para la conexión a la base de datos y la inserción de la información, se ha utilizado la librería pymongo.

Los parámetros de la conexión se van a especificar en el fichero de configuración configuración.ini que fue creado previamente para almacenar el tiempo entre muestras de gas. Al comienzo del programa se carga toda esta

información como el host, puerto, usuario, contraseña y nombre de la base de datos y de la colección.

Por otra parte, hay que completar la comunicación entre las placas y la aplicación. Para ello, la aplicación se va a suscribir a los topics en los que las placas con los sensores van a publicar los datos. En principio se tendría:

- mq135/mesa, mq135/estantería, para los sensores de gas, indicando su localización.
- giroscopio/puerta, giroscopio/ventana, para los sensores que van a medir los ángulos de apertura y cierre.

Además de ello, la aplicación debe enviar por MQTT la frecuencia de medición que contenga el fichero de configuración, para lo cual enviará los datos a los topics configuracion/mq135/estantería y configuracion/mq135/mesa, de donde después, las placas correspondientes consumirán la información.

De esta manera, el funcionamiento sería el siguiente:

- Al principio, se lee el fichero de configuración y se envía la frecuencia de muestreo a los sensores de gas.
- Mientras tanto, los sensores de la puerta y ventana determinarán los estados en los que se vayan encontrando estos elementos y los enviarán. La aplicación mantendrá unas variables para conservar dichos estados actuales.
- Cuando cada uno de los sensores de gas envíe datos, la aplicación creará un paquete con la información de puerta y ventana, el dato en ppm y el dato en raw de dicho sensor y lo enviará al servidor MongoDB.

Pruebas realizadas.

Las pruebas realizadas se basan en poner en marcha los diferentes componentes del sistema para comprobar que los datos se envían correctamente y se almacenan en la base de datos tal y como se especifica.

2.2.5. Quinta iteración. Detección de personas.

En este punto se añade la funcionalidad para estimar el número de personas que se encuentran en la estancia en cada momento para así tener aún más información del contexto.

Análisis de requisitos.

Los requisitos que se encuentran son:

- Debe poder detectar personas que dispongan de un dispositivo móvil.

Búsqueda de información.

Para detectar y hacer el recuento de las personas que se encuentran en la estancia de forma sencilla, se ha optado por implementar un sniffer, que permite analizar el tráfico de la red. El funcionamiento de este programa se basa en la escucha de los paquetes que los dispositivos conectados a la red (o no) envían casi constantemente. Por ejemplo, los smartphones realizan búsquedas de los puntos de acceso que hay a su alrededor para intentar encontrar puntos de acceso conocidos a los que se haya conectado con anterioridad. Este tipo de paquetes se conoce como probe request. Gracias a estos paquetes (que son un conjunto de bytes), se puede obtener información relevante como la dirección MAC de los dispositivos o la intensidad de la señal de los paquetes que se envían (Received Signal Strength Indicator, RSSI).

Gracias a esta información, se puede realizar una estimación de las personas que se encuentran más cercanas y más lejanas al punto donde se ubica el sniffer, de forma que la intensidad será mayor cuanto más cerca se encuentre y viceversa. Así, se puede discernir si una persona se encuentra dentro o fuera de una habitación, teniendo en mente que dentro de la misma los valores de intensidad deben ser mayores que fuera de la misma.

En este punto, se decidió probar diferentes proyectos encontrados en la red para comprobar su funcionamiento, ya que el proceso para separar las diferentes partes de los paquetes y extraer los bytes necesarios es un poco

tedioso. Cabe destacar que la gran mayoría de trabajos se basan en el código de Ray Burnette.

Finalmente, tras diferentes pruebas, se ha optado por utilizar el trabajo de Andreas Spiess [20], debido a que su trabajo también envía los datos que recoge. Sin embargo, es conveniente realizar modificaciones para ajustarlo a las necesidades de nuestro caso de uso.

Su funcionamiento se basa en el modo promiscuo que provee la API del SDK del ESP8266 (el procesador que se está utilizando) y permite capturar paquetes del estándar IEEE 802.11b/g/n. Mediante un callback que recibe un conjunto de bytes y una longitud de paquete, se puede deducir qué tipo de paquete es y extraer información relevante del mismo como el emisor y el destino.

Diseño.

Dado que este programa se va a ejecutar desde el microcontrolador NodeMCU, la estructura del código viene dada por las funciones setup y loop, además del callback encargado de procesar los paquetes. En setup se configuran los parámetros necesarios para que comience a ejecutarse el bucle principal. En el loop, se comprueban ciertos parámetros (como el número de direcciones recogidas o el tiempo) para enviar los datos a la aplicación sumidero. Mientras tanto, el callback queda a la escucha de paquetes para poder extraer la información de estos.

Estos tres métodos principales se sustentan en otros métodos y estructuras que se encuentran organizados en los ficheros functions.h y structures.h, respetivamente, que permiten desgranar los paquetes y obtener la información de los diferentes campos.

Aunque el funcionamiento del sketch es correcto, es necesario añadir un filtrado de direcciones MAC, debido a que el rango en el que este dispositivo puede captar paquetes es muy amplio y a que los demás dispositivos que se

encuentran en la estancia midiendo, enviando y recibiendo datos también son detectados, aunque no son significativos, pues no pertenecen a una persona.

El funcionamiento que sigue el programa es el siguiente:

- El método loop va alternando los canales sobre los que trabaja el estándar 802.11b/g/n. En cada canal permanece un tiempo determinado (200 ms).
- Mientras tanto, el callback permanece a la escucha recibiendo paquetes y almacenando la información que contienen en estructuras de datos. Para ello hace uso de las estructuras que conforman los campos de los diferentes paquetes y va transformando los bytes a otro tipo de datos más manejables. Todo ello mediante los métodos y estructuras de los ficheros functions.h y structures.h.
- Finalizado el recorrido por los 14 canales, comprueba las estructuras que ha ido utilizando el callback para comprobar cambios y enviar los datos si hubiera cambios. Además, pasado un tiempo, realiza una limpieza de las estructuras llamando a otro método.
- Por otra parte, cuando se detectan cambios en el número de direcciones MAC de los paquetes recibidos, se procede a enviar esa información mediante MQTT. Simplemente se comprueba la intensidad de la señal para que haga referencia a un dispositivo que se encuentra dentro de la estancia y no fuera. Además, se comprueba que no corresponde a ninguna de las MACs conocidas de los demás dispositivos de la habitación. Una vez que se tienen las direcciones “correctas”, se hace un recuento y se envía ese número.
- Otra función relevante es purgeDevice, que se encarga de eliminar los dispositivos que hace un tiempo que no se encuentran. Para ello se utiliza un parámetro de tiempo que hace referencia a la última vez que fueron “descubiertos” por el callback. Esto nos permite actualizar el número de personas que se encuentran en la estancia cada cierto tiempo.

Implementación.

En este apartado se va a hacer énfasis en los cambios que se han añadido sobre el trabajo de partida.

En primer lugar, se ha creado una estructura de datos conformada por las direcciones MAC conocidas de dispositivos que no hacen referencia a ninguna persona, por lo que no son útiles para obtener información. Para ello se ha creado un set de string que va a almacenar todas estas direcciones como claves de la estructura, llamado macs_excluidas. En este punto, había que identificar qué dispositivo tenía cada dirección, para lo cual se consultó la información que provee la interfaz del router en la sección de clientes DHCP (Figura 28), aprovechando que todos estos dispositivos estaban conectados a la misma red.

ESP-3C906A	192.168.1.120	80:7D:3A:3C:90:6A	23:52:45
ESP-3EAA5B	192.168.1.122	80:7D:3A:3E:AA:5B	23:59:15
ESP-C521CC	192.168.1.109	EC:FA:BC:C5:21:CC	23:59:37

Figura 28. Captura interfaz del router con direcciones MAC.

Este conjunto de direcciones almacenado en el fichero structures.h va a ser consultado para filtrar las direcciones que realmente sí pertenecen a personas. Este filtrado se realiza en el momento previo al envío de los resultados, mediante el método find que provee la estructura de datos set. De esta forma, si en el array que ha ido rellenando el callback durante el escaneo previo se encuentra una dirección “no válida”, no se contabiliza para enviarla posteriormente a la aplicación sumidero.

Por otra parte, hay que discernir entre los dispositivos que se encuentran dentro de la habitación o fuera, debido a que el sniffer es capaz de capturar paquetes con diferente fuerza en la señal. Para ello, teniendo en cuenta que esta fuerza se mide en decibelios (en negativo), se puede deducir que cuanto menor sea el número, más cerca estará el dispositivo del sniffer, ya que mayor será la intensidad con la que recibe los paquetes. En este punto, mediante la realización de pruebas con un smartphone con MAC conocida, se selecciona

un umbral sobre el cual discernir si el dispositivo está dentro o fuera. Hay que tener en cuenta la colocación del sniffer en el centro de la estancia para que la distancia sea máxima desde todos los puntos de la habitación, como si del radio de una circunferencia se tratara.

Para ello, se aprovecha el dato que se obtiene de los paquetes capturados llamado RSSI (Received Signal Strength Indicator). Se crea una variable con el valor para umbralizar los datos y teniendo el array de todos los dispositivos encontrados, se filtra mediante su RSSI. En este caso, el valor sobre el cual se puede decir que un dispositivo está dentro es -50 (y menores), aunque dependiendo del tamaño de la estancia puede variar.

Sin embargo, con las pruebas realizadas durante esta implementación, se descubre que hay dispositivos que a veces mandan paquetes con una alta intensidad de señal pero que no se encuentran ni siquiera dentro de la casa. Esto hace que se generen picos, ya que se detectan sólo algunas veces. Para intentar paliar este efecto, se ha ideado una función en la que se detecte una intensidad media medida entre diferentes ejecuciones del sniffer. De esta forma un valor puntual muy bajo no será tan bajo comparado con los valores que ha tenido en las demás ejecuciones consideradas.

Finalmente, una vez que se realiza este filtrado de distancia y direcciones excluidas, se envía la información mediante MQTT haciendo uso de la librería PubSubClient y formateando el mensaje como JSON mediante la librería ArduinoJson. En este caso, solo se envía un campo llamado afluencia, y el contador de direcciones MAC válidas. Este dato se envía al topic al que también se suscribirá la aplicación, llamado sniffer/1. Los cambios en dicha aplicación son mínimos.

Pruebas realizadas.

Las pruebas realizadas consisten en colocar dispositivos móviles en la estancia y poner en marcha el sniffer para comprobar que detecta el número correcto de dispositivos.

2.2.6. Sexta iteración. Plataforma IoT.

En este punto se pretende hacer uso de una plataforma que permita visualizar la información de manera gráfica que se va produciendo en tiempo real.

Análisis de requisitos.

Los requisitos que se encuentran son:

- Se debe poder visualizar la información recogida en tiempo real de la estancia.
- Debe poder ejecutarse en Raspberry Pi 3, dispositivo que se utilizará posteriormente.
- Debe ser usable e intuitivo.

Búsqueda de información.

En primer lugar, se realizó una búsqueda de plataformas IoT open source más utilizadas para revisar las opciones más relevantes del momento. En primer lugar, se consideró Thinger.io, debido a que ofrece un servicio cloud en el que manejar los datos (casi) sin tener en cuenta el almacenamiento. Sin embargo, posee una restricción que hace que no sea viable para nuestro caso de uso: el número de dispositivos que se pueden conectar se limita a dos (Figura 29).

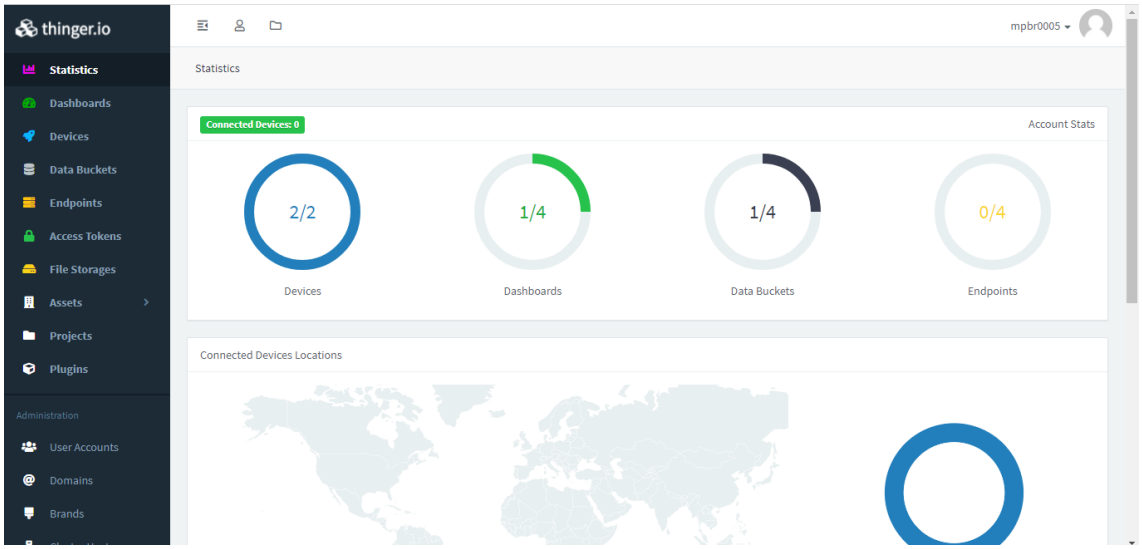


Figura 29. Interfaz de thinger.io.

A continuación, se optó por Thingsboard. En un primer momento se instaló en el ordenador para realizar pruebas y comprobar su funcionamiento.

Finalmente, debido a las características que incluye, se opta por esta opción. Además de ello, permite la instalación en Raspberry.

Desarrollo.

En primer lugar, para poder alojar nuestra propia instancia de ThingsBoard en Windows, es necesario seguir los pasos que aparecen en la web oficial. El primer requisito es instalar Java 11. Seguidamente es necesario descargar la base de datos sobre la que trabaja ThingsBoard. En este punto se puede elegir un SGBD SQL (recomendado para aplicaciones con menos requerimientos) o híbrido (para entornos profesionales). En este caso se opta por la primera opción, que es la recomendada para este caso de uso. Para ello, se instala y configura PostgreSQL, tal y como se especifica en la documentación de ThingsBoard.

A continuación, se descarga e instala el propio software de ThingsBoard, disponible en GitHub. Por último, se pone en marcha el servicio mediante la orden `net start thingsboard`. Para acceder a la aplicación se introduce

http://localhost:8080/ en el navegador y aparecerá una pantalla de login como en la Figura 30.

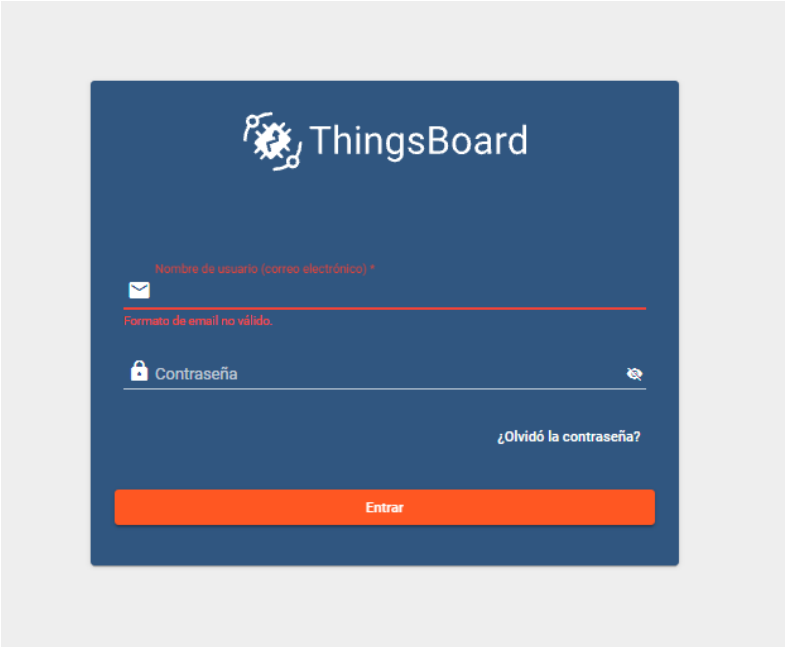


Figura 30. Interfaz login ThingsBoard.

Para iniciar sesión la primera vez, ThingsBoard viene configurado con un correo y contraseña por defecto que posteriormente puede cambiarse. Una vez “dentro” se visualizan todas las opciones que incluye según cada rol:

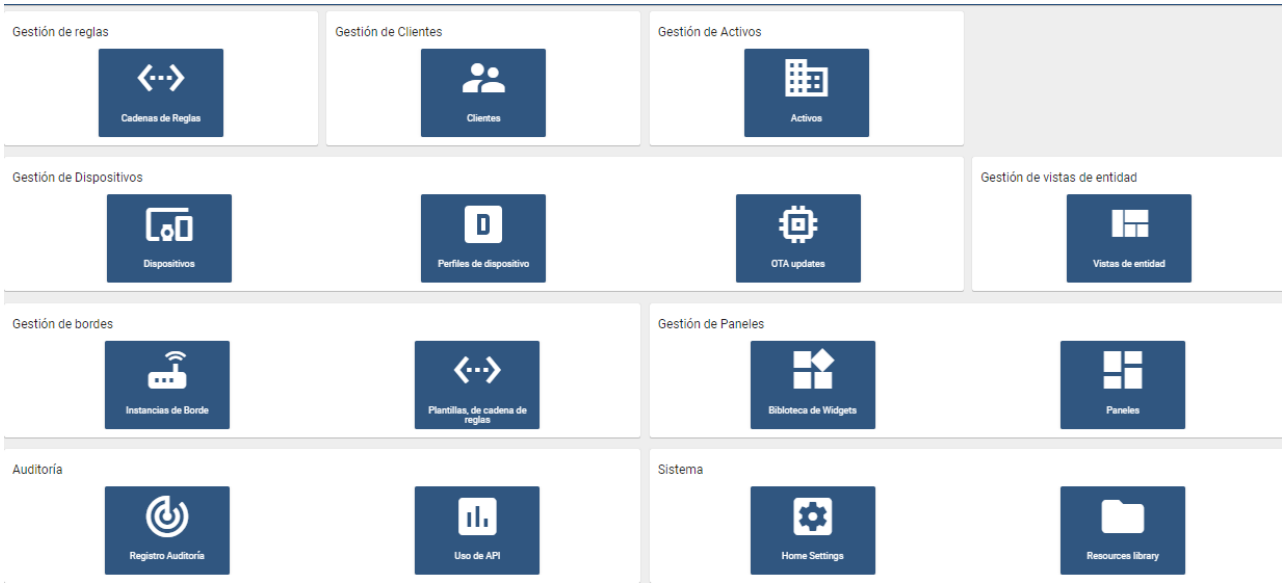


Figura 31. Vista principal rol de administrador.

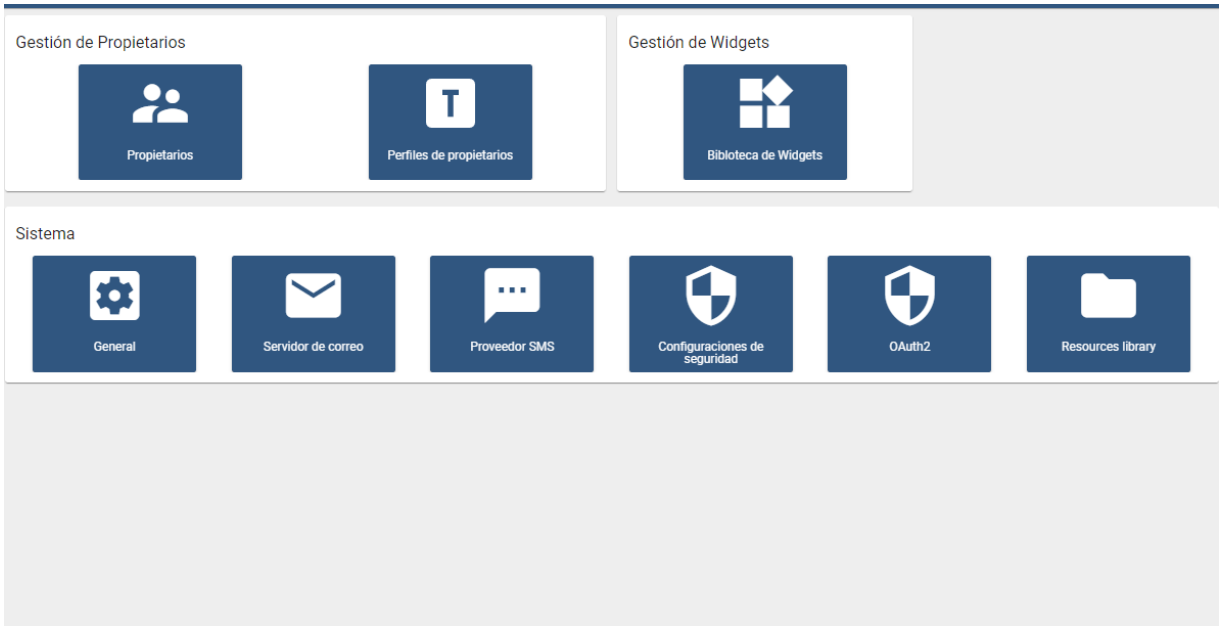


Figura 32. Vista principal rol de administrador principal.

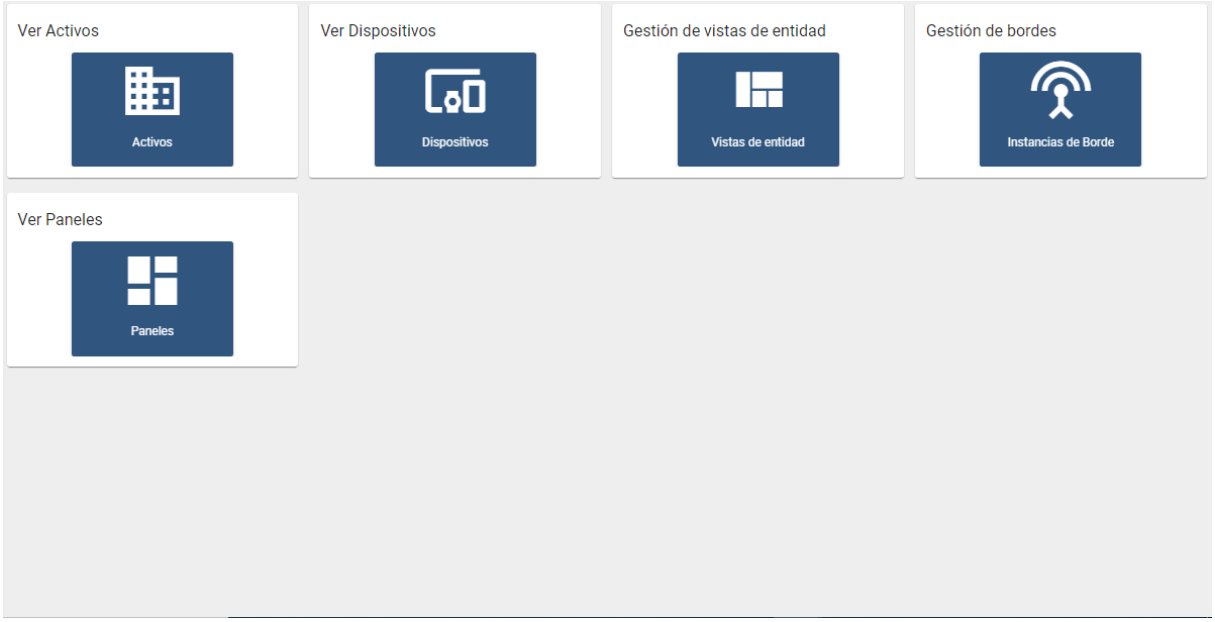


Figura 33. Vista principal rol de cliente.

Esta plataforma ofrece comunicación mediante el protocolo MQTT, idóneo para estos pequeños dispositivos que se utilizan. Por ello se decide que sea la forma de comunicación. Para conectar mediante MQTT es necesaria la creación de dispositivos que van a tener un id y un token que actúan como usuario y contraseña a la hora de enviar los mensajes. En Arduino es muy sencillo este cambio, ya que únicamente hay que añadir estos dos parámetros al utilizar la librería PubSubClient.

Todos los datos de todos los dispositivos se envían al mismo topic: `v1/devices/me/telemetry` y, en función del id del dispositivo, ThingsBoard asocia estos mensajes a cada uno de ellos internamente. Es imprescindible que estos mensajes tengan formato JSON, ya que de ahí se extraen los atributos que luego se van a utilizar en la plataforma.

El siguiente paso es crear un tablero o dashboard en el que poder mostrar la información que va llegando de los dispositivos. En este punto se puede importar directamente un tablero o crearlo desde cero, por lo que se muestra una de las ventajas de ThingsBoard: poder exportar el trabajo que se ha creado para replicarlo en otra instancia.

Dentro de cada tablero se incluyen los widgets, que son los elementos que van a mostrar la información de forma gráfica. Esta es otra de las ventajas de esta plataforma, la gran cantidad de widgets (Figura 34) de diferentes tipos que se encuentran a disposición para poder representar casi cualquier tipo de información.

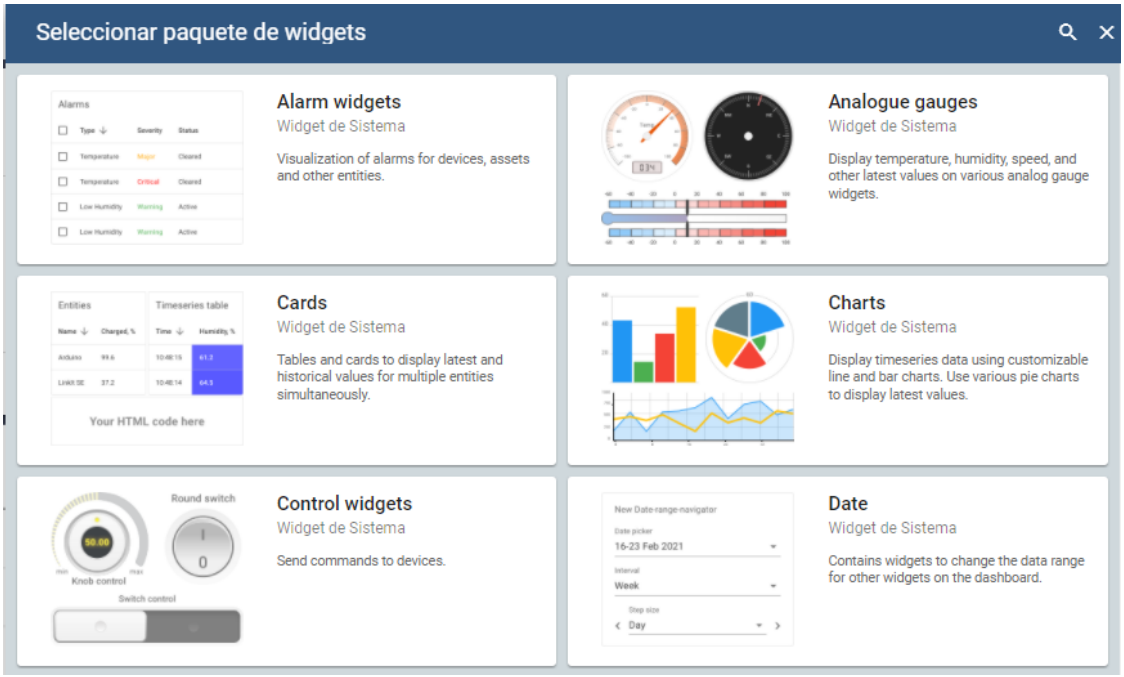
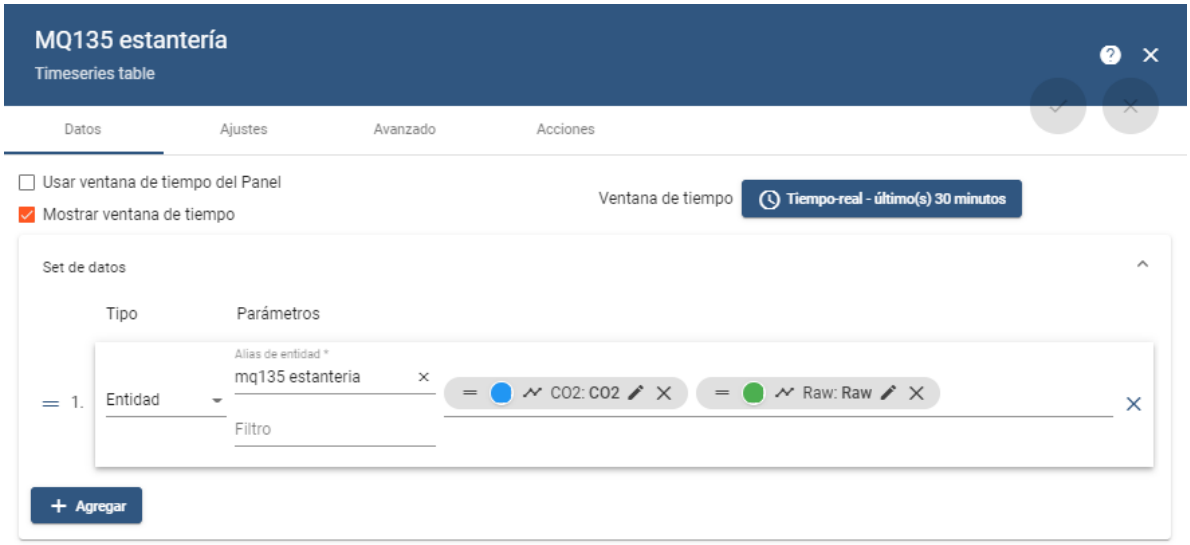


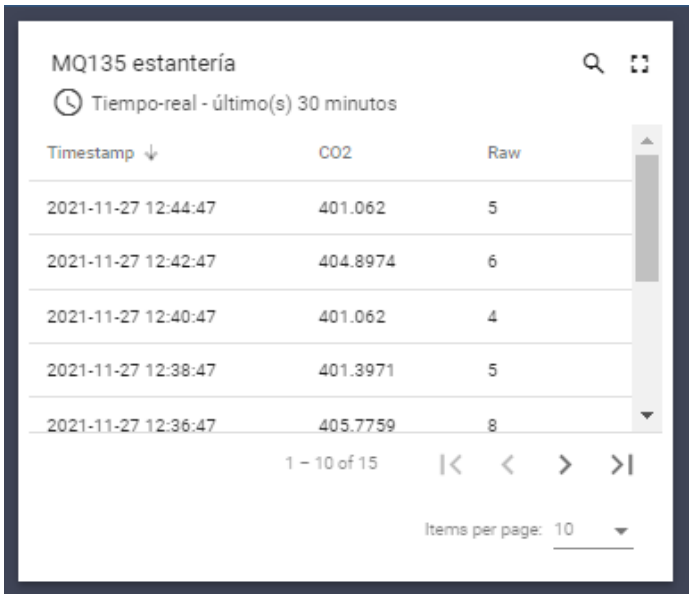
Figura 34. Algunos de los widgets disponibles en ThingsBoard.

En este caso se necesita representar:

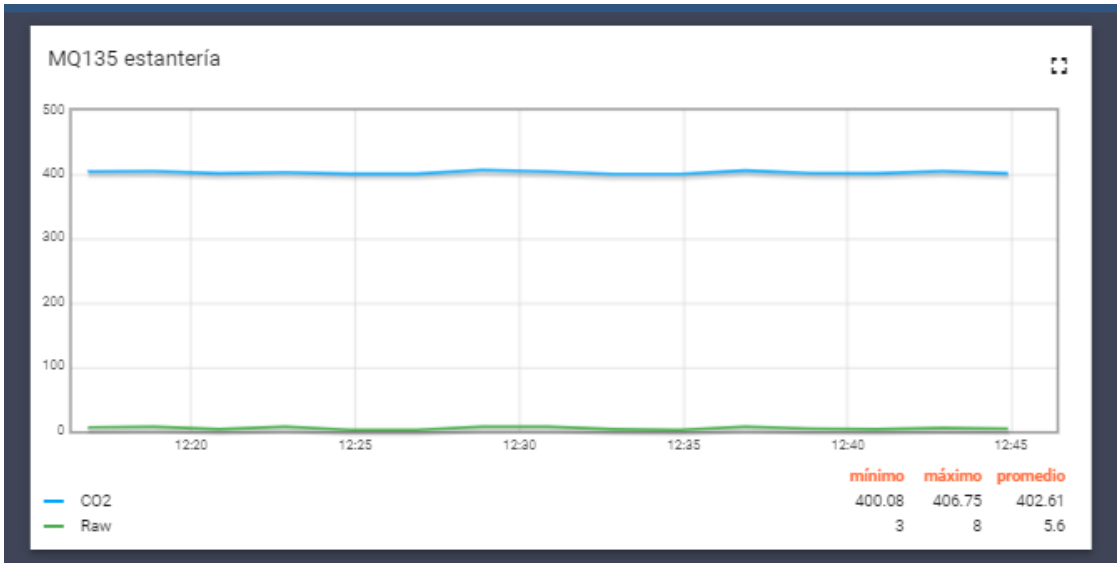
- Datos de los sensores de gas
- Estado de puerta y ventana
- El número de personas que se encuentra en la estancia

Para mostrar los datos de los sensores de gas, se decide utilizar una tabla en la que se muestran los valores que va recibiendo de los mensajes. Este widget se denomina *Timeseries table.*





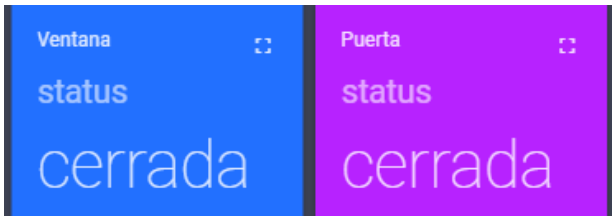
Además de visualizar los datos de forma numérica, se decide añadir otro widget que resuma todos estos datos y dé una visión más amplia del ambiente en cuanto al gas. Para ello, de los muchos widgets que incorpora ThingsBoard de tipo gráficas, se elige el denominado *Timeseries Line Chart,* una gráfica de líneas en la que cada punto corresponde a un valor en el tiempo y está unido a los valores anteriores y siguientes.

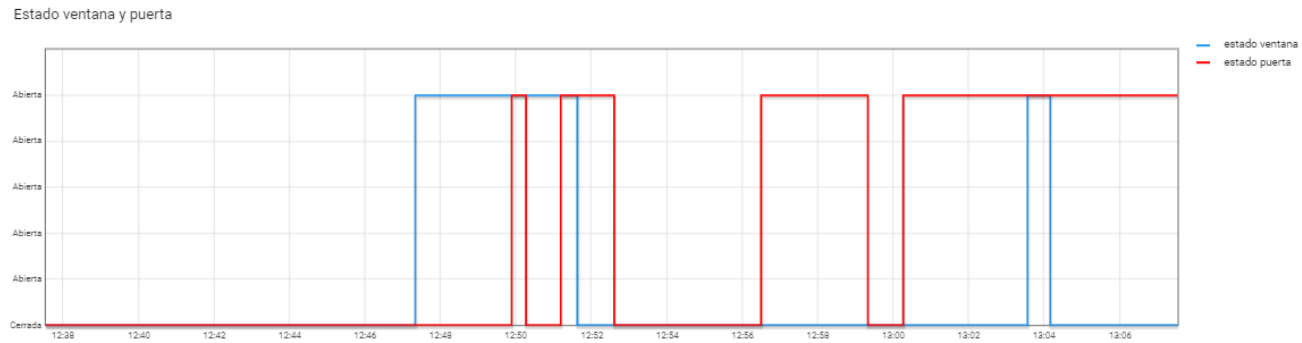


Para ambas gráficas se usan los mismos datos del sensor de gas: nivel de partículas de CO_2 y datos raw.

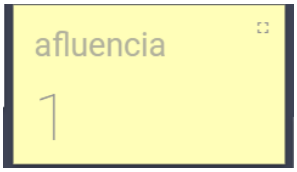
Al igual que para un sensor MQ135 se hace lo mismo para el otro que está colocado en otra ubicación.

Para representar la información recogida de la puerta y la ventana, se ha optado por dos tipos de widget: *Simple Card* y *State Chart*. El primero de ellos indica el estado de la puerta/ventana en cada momento actualizado y el segundo muestra un histórico del estado de ambas.



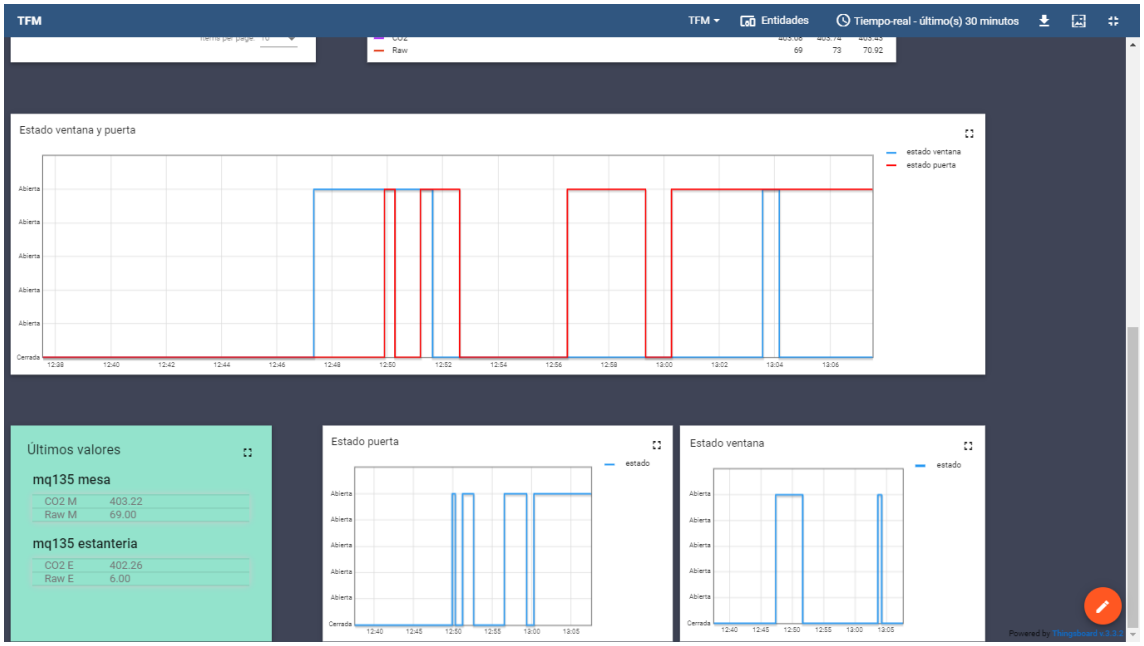


Por último, para representar el número de personas, se ha optado por usar un widget *Simple Card* como en el caso anterior.



Pruebas realizadas.

Las pruebas que se han realizado consisten en la puesta en marcha de los dispositivos de medición y recogida de datos y a la vez ThingsBoard para comprobar que se actualiza la información de forma correcta.



2.2.7. Séptima iteración. Migración a Raspberry.

Hasta ahora, tanto la aplicación de recogida de datos como la ejecución de la plataforma IoT se realizaban en el propio ordenador. En este momento se incluye la Raspberry a modo de servidor, en el que se aloje la instancia de ThingsBoard y la aplicación “sumidero” desarrollada anteriormente. De este modo no es necesario destinar un equipo de trabajo a estas tareas.

Análisis de requisitos.

El requisito que se encuentra es:

- Debe poder funcionar sin necesidad de interactuar con el sistema, únicamente conectándolo a la corriente.

Desarrollo.

La migración es un proceso sencillo, ya que la distribución instalada en la Raspberry (Raspberry Pi OS) incluye Python, el lenguaje en el que se ha desarrollado la aplicación de escritorio. Por lo tanto, lo único que hay que hacer

es trasladar la carpeta con los archivos del programa a la Raspberry, por ejemplo, usando uno de los puertos USB de los que dispone.

Sin embargo, también hay que encontrar la manera en la que se automatice la ejecución de la aplicación, para que al conectarla a la alimentación se ejecute directamente sin necesidad de realizar ninguna acción. Para ello, se ha optado por el uso de archivos systemd, tras probar otras opciones como el comando crontab. La principal ventaja de este método es que se puede especificar el momento en el que se va a ejecutar y las partes que se requieren inicializadas antes de la ejecución de nuestra aplicación, entre otros parámetros que se pueden detallar.

En primer lugar, se crea un fichero Unit llamado IoT.service mediante el comando `sudo nano /lib/systemd/system/IoT.service` como se muestra en la Figura 35.

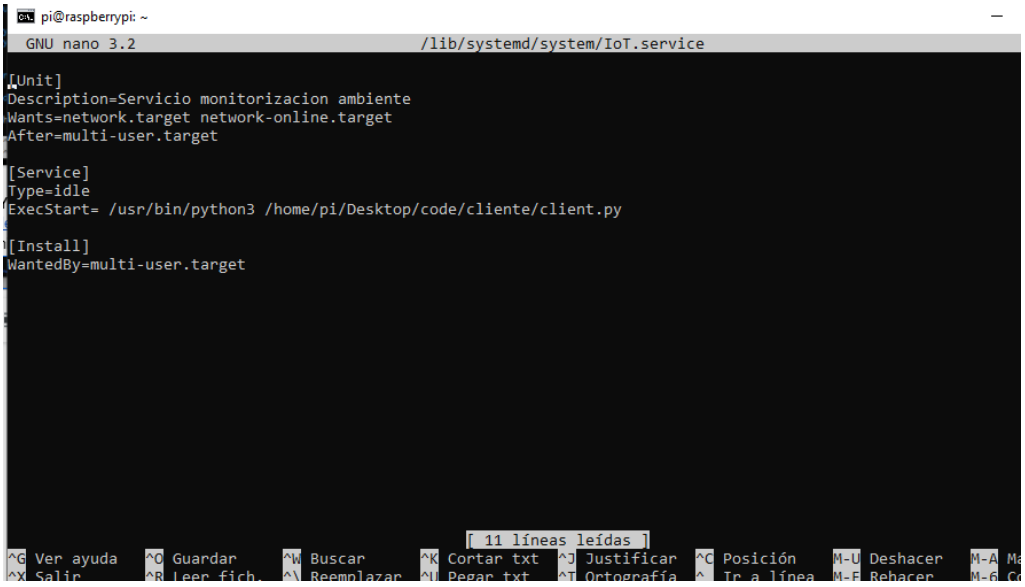


Figura 35. Contenido del archivo IoT.service.

En la sección [Unit] se describe el servicio y se especifica la necesidad de tener cargada la parte de red antes de la ejecución, puesto que nuestra aplicación va a comunicarse mediante MQTT con los dispositivos y va a subir los datos a una base de datos remota, tareas para las cuales necesita conectividad de red. En la parte [Service] se indica la ruta absoluta hasta la aplicación que se va a

ejecutar. La última parte [Install] especifica el nivel de ejecución en la inicialización (runlevel) que ha de ser alcanzado antes de ejecutar la aplicación.

A continuación, es necesario dar permisos de ejecución al archivo con `sudo chmod 644 /lib/systemd/system/IoT.service` y ponerlo en marcha con los comandos `sudo systemctl daemon-reload` y `sudo systemctl enable IoT.service` para que cada vez que se encienda la Raspberry comience a ejecutarse la aplicación.

Una vez configurada esta parte, hay que instalar ThingsBoard en la Raspberry para iniciar una instancia y poder importar la configuración realizada anteriormente de dashboard y widgets. Este proceso se explica detenidamente en el anexo ThingsBoard en Raspberry Pi model 3.

Debido a que ThingsBoard incluye un propio broker MQTT, los dispositivos tendrían que realizar dos publicaciones en diferentes broker, para evitar esto y optimizar el envío de datos, se puede usar ThingsBoard IoT Gateway, que permite la integración con servicios externos. De esta forma, los dispositivos publican en el broker "nativo" ejecutado en Raspberry y ThingsBoard actúa a como cliente suscribiéndose a estos topics y recibiendo los mismos mensajes que recibe la aplicación de escritorio. La instalación y configuración se describe en los anexos ThingsBoard IoT Gateway en Raspberry Pi model 3. y Configuración ThingsBoard IoT Gateway.

Una vez realizada la instalación de ambos servicios, en el momento en que los dispositivos comiencen a publicar en los respectivos topics, ThingsBoard automáticamente creará los perfiles de estos dispositivos en la plataforma con los parámetros que aparezcan en el mensaje según la estructura especificada.

Pruebas realizadas.

Las pruebas realizadas se basan en poner en marcha los dispositivos y la Raspberry y comprobar que se visualizan los datos correctos en ThingsBoard y que la aplicación sumidero funciona con normalidad.

3. Experimentación, resultados y discusión.

La experimentación se va a basar en la observación del progreso de las variables de gas (niveles de CO_2 y datos raw) en diferentes condiciones con el paso del tiempo, teniendo en cuenta:

- Número de personas en la estancia en cada instante.
- Apertura de ventana en cierto momento.
- Apertura de puerta en cierto momento.
- Observar las posibles diferencias entre el medidor que se encuentra más cercano a la ventana y el que está junto a la puerta.

Las gráficas que se van a comentar siempre van a tener en el eje de abscisas la dimensión temporal, y en el eje de ordenadas, la variable que se va a observar.

Primer caso de estudio

En este primer caso se ha realizado una sesión en la que participan dos personas durante aproximadamente dos horas. Se comienza a medir gas desde las 11:53 y se finaliza a las 14:10. A continuación se muestran las capturas de ThingsBoard.

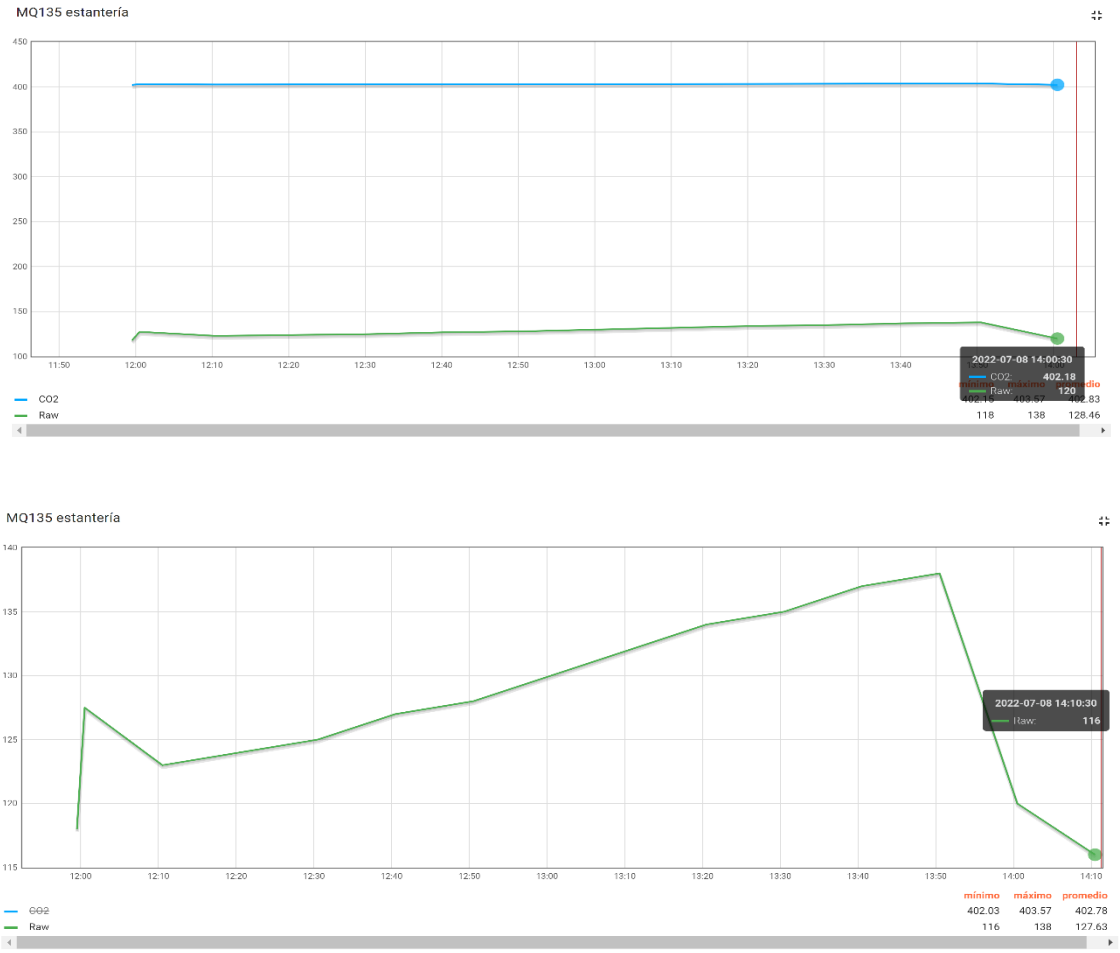


Figura 36. Primer caso de estudio. Gráfica gas estantería.

En primer lugar, se van a comentar las gráficas del sensor de gas ubicado en la esquina de la habitación junto a la puerta. En la primera gráfica se muestran los valores conjuntos del CO_2 que proporciona la librería (azul) y los valores raw que recoge el sensor (verde). Lo primero que se puede observar es el primer pico que se encuentra en la gráfica, que se corresponde con la calibración del sensor junto a la ventana alimentado por una batería y, después, introducido en la estancia. Justo en la siguiente medición ya se estabilizan los valores, recordando que, este tipo de sensores funcionan a una temperatura elevada y, a veces, tardan un poco en llegar hasta esta. Así, en dicha gráfica, no se observa una gran diferencia de CO_2, la línea azul se mantiene casi estática (en torno a 402-403 ppm) y la verde sí muestra cambios más significativos.

En cambio, en la gráfica siguiente, en la que sólo se seleccionan los valores raw, se observa la subida progresiva de los valores, desde un mínimo situado

en torno a 123, hasta un máximo situado en torno a 138 a las 13:50, cuando el dispositivo llevaba midiendo dos horas sin ningún tipo de ventilación. Después de ese momento, los valores comienzan a disminuir bruscamente justo después de que la puerta y la ventana se abrieran (14:00) y llegando al siguiente mínimo, 116, en la última medida tomada a las 14:10 tras un breve periodo de ventilación.

A continuación, se observa los valores que ha ido midiendo el sensor, tanto de CO_2 como raw.

MQ135 estantería

Timestamp	CO_2	Raw
08/07/2022 11:59:53	402,1459656	118
08/07/2022 12:00:03	402,7229614	127
08/07/2022 12:00:13	402,7580261	128
08/07/2022 12:10:13	402,4218445	123
08/07/2022 12:20:13	402,5193481	124
08/07/2022 12:30:14	402,6196899	125
08/07/2022 12:40:14	402,6882019	127
08/07/2022 12:50:14	402,7934265	128
08/07/2022 13:00:14	402,9017029	130
08/07/2022 13:10:14	403,1274414	132
08/07/2022 13:20:14	403,2055359	134
08/07/2022 13:30:15	403,3253174	135
08/07/2022 13:40:15	403,4484253	137
08/07/2022 13:50:15	403,5748901	138
08/07/2022 14:00:15	402,175415	120
08/07/2022 14:10:15	402,0310364	116

Tabla 4. Primer caso de estudio. Valores gas estantería.

Para este sensor, el valor mínimo ha sido 116 y el máximo 138, lo que hace una diferencia de 12 puntos en dos horas con dos personas en el interior en un intervalo de 137 minutos.

Ahora se procede a revisar las gráficas del sensor de gas ubicado en el otro extremo de la estancia, en la mesa junto a la ventana.

Figura 37. Primer caso de estudio. Gráfica gas mesa.

En este caso, como ocurría con el sensor anterior, la primera gráfica en la que se representan tanto los valores CO_2 como los valores raw, es complicado comprobar el comportamiento, por lo que se pasa a comentar sólo las variaciones raw, que son más significativas que los valores que proporciona la librería. Al principio, esta línea roja representa un valor de 76, que posteriormente baja hasta 69 una vez estabilizado y alcanzada una mayor temperatura. Después los valores medidos van subiendo poco a poco hasta las 12:54, que se alcanza un pico en la medición. Ese pico se puede considerar un valor anómalo, ya que no hay ningún cambio en las condiciones de medición de la habitación. La siguiente medición ya muestra un valor más real a la tendencia que había hasta las 12:44. A continuación, se registra otro pico, pero

esta vez puede considerarse más “normal”, debido a que la diferencia entre los valores es mínima. Tras esto, los valores siguen ascendiendo poco a poco hasta llegar al máximo global, 89 (13:44), tras el cual los valores comienzan a disminuir bruscamente debido a la apertura de ventana y puerta.

En este caso, los valores han oscilado entre 66 y 89 (23 puntos de diferencia) en 137 minutos.

En la siguiente tabla se muestran los valores numéricos que ha ido registrando el sensor durante la ejecución de esta experimentación.

MQ135 mesa

Timestamp	CO_2	Raw
08/07/2022 11:53:47	401,9945068	72
08/07/2022 11:53:57	402,9333801	80
08/07/2022 11:54:07	402,1261902	72
08/07/2022 12:04:07	401,8279419	69
08/07/2022 12:14:07	401,9519043	70
08/07/2022 12:24:07	402,1261902	72
08/07/2022 12:34:08	402,1714172	74
08/07/2022 12:44:08	402,5577393	78
08/07/2022 12:54:08	403,1646729	84
08/07/2022 13:04:08	402,8223572	79
08/07/2022 13:14:08	402,9333801	80
08/07/2022 13:24:08	402,8775024	79
08/07/2022 13:34:08	403,1646729	82
08/07/2022 13:44:09	403,9358521	89
08/07/2022 13:54:09	402,4076538	76
08/07/2022 14:04:09	401,5602112	66

Tabla 5. Primer caso de estudio. Valores gas mesa.

A continuación, puede visualizarse el estado de la puerta y ventana durante toda la sesión. Estos dispositivos se ponen en marcha antes de comenzar a calibrar y medir los datos de gas, para observar el estado anterior en el que se encontraba la estancia. Se observa que, al principio se abren tanto la puerta como la ventana, después se cierran cuando los dispositivos ya se han calibrado y se mantienen cerradas durante casi dos horas. A las 13:54 se vuelven a abrir para ventilar la estancia hasta el final de la sesión.

Figura 38. Primer caso de estudio. Apertura puerta y ventana.

Por último, se revisa la gráfica de la afluencia, medida mediante el sniffer. Aquí sólo ha habido dos personas en la habitación, aunque se han estado registrando desde 0 hasta 4 personas. Hay que recordar que este dispositivo capta los paquetes que son enviados de forma inalámbrica mediante la tecnología Wifi, no solo de la red de la casa. Comprobando los resultados de otras ejecuciones, se puede decir que un dispositivo que esté relativamente cercano puede dar valores más bajos que otro que esté más lejano, debido a que hay algunos paquetes que se envían con una mayor intensidad. Es por eso que hay valores superiores a 2. Además, hay veces en las que un dispositivo no envía datos con tanta frecuencia, por lo que, considerando una ventana temporal, los valores no llegan a actualizarse y ese dispositivo queda “fuera” de la habitación. Por este motivo, hay veces que la afluencia es menor que 2.

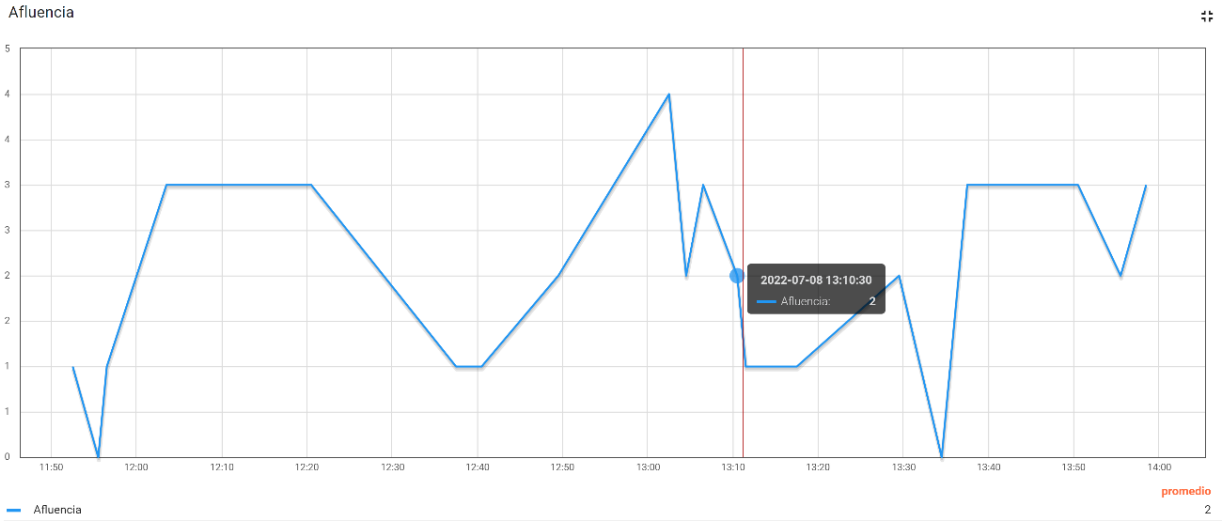


Figura 39. Primer caso de estudio. Afluencia.

Segundo caso de estudio

En este caso se realiza una sesión en la que participan tres personas durante poco más de dos horas. Se comienza a medir gas desde las 12:12 y se finaliza sobre las 14 horas. A continuación, se muestran las capturas de ThingsBoard.

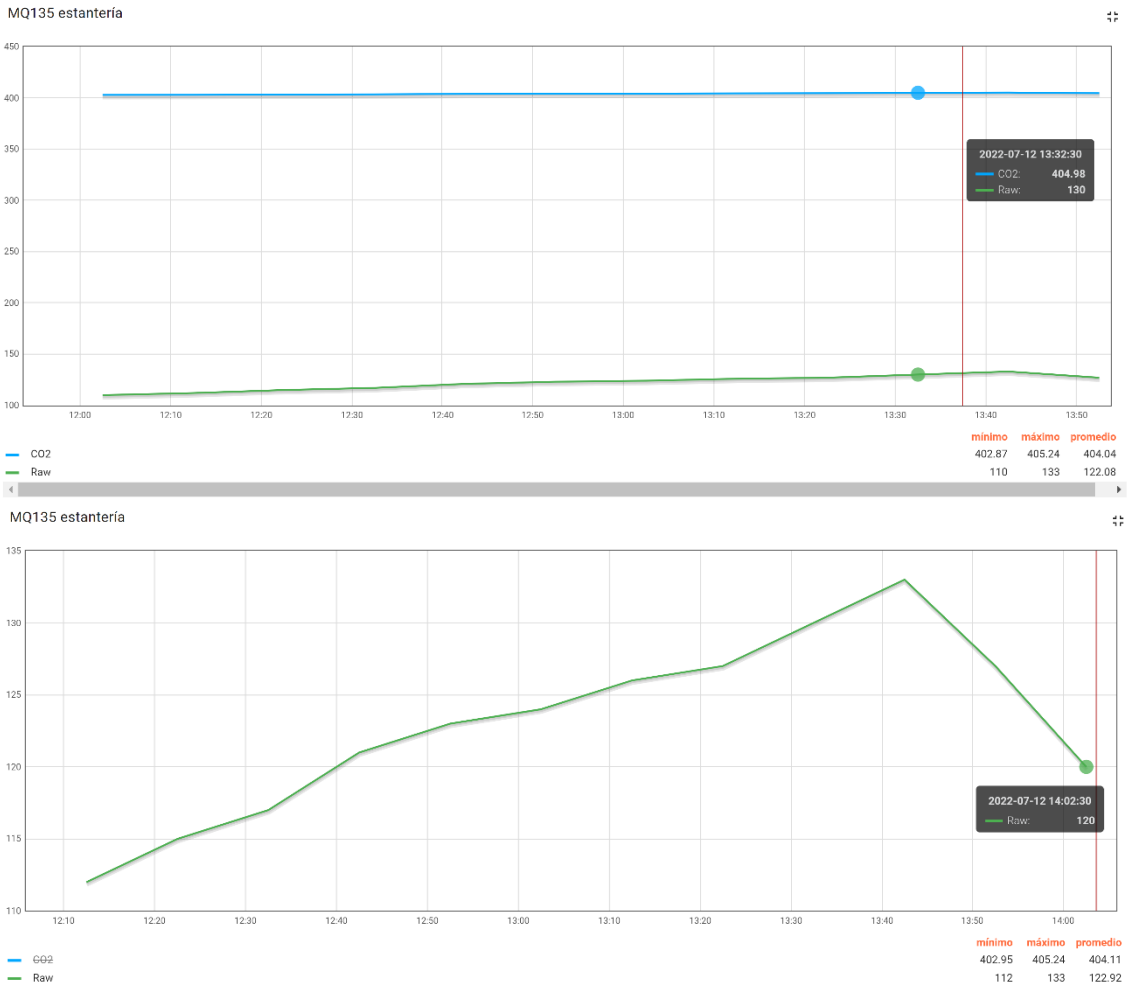


Figura 40. Segundo caso de estudio. Gráfica gas estantería.

Se va a comentar las gráficas del sensor de gas en la estantería. Al igual que en el caso anterior, se analizan los valores raw ya que la librería no muestra los cambios lo suficientemente destacados (de 401 a 404). Se aprecia una subida lineal desde los primeros valores medidos hasta el máximo a las 13:42 y después una caída lineal de estos debida al inicio de la ventilación. Aquí se observa de forma clara que la pendiente cuando los valores son crecientes es mucho menor que cuando estos bajan debido a la apertura de puerta y

ventana. Esta vez el valor mínimo ha sido 112 y el máximo 133 (diferencia de 21 puntos) que se han dado en un tramo de 90 minutos.

En la siguiente imagen se muestran los valores numéricos obtenidos por este sensor.

MQ135 estantería

Timestamp	CO_2	Raw
12/07/2022 11:52:00	401,9880371	98
12/07/2022 11:52:10	402,8263855	109
12/07/2022 11:52:20	402,7854309	109
12/07/2022 12:02:21	402,8678284	110
12/07/2022 12:12:21	402,9519653	112
12/07/2022 12:22:21	403,3066406	115
12/07/2022 12:32:21	403,4951782	117
12/07/2022 12:42:22	403,8956604	121
12/07/2022 12:52:22	404,0541687	123
12/07/2022 13:02:22	404,1624451	124
12/07/2022 13:12:22	404,3853455	126
12/07/2022 13:22:23	404,5581970	127
12/07/2022 13:32:23	404,9810486	130
12/07/2022 13:42:23	405,2353210	133
12/07/2022 13:52:24	404,5581970	127
12/07/2022 14:02:24	403,7925415	120
12/07/2022 14:12:24	402,3988953	104

Tabla 6. Segundo caso de estudio. Valores gas estantería.

Seguidamente se analiza la gráfica generada con la información del sensor ubicado en la mesa junto a la ventana.

Figura 41. Segundo caso de estudio. Gráfica gas mesa.

Fijándonos en la gráfica en la que sólo se representan los valores raw se intuye que tiene una forma similar a la gráfica que representa los valores del sensor de la estantería. Como diferencia, esta gráfica presenta algunos valores que se "salen" un poco de la tendencia, aunque finalmente, en conjunto, son valores representativos. El mínimo global (64) aparece al principio de la ejecución, y el máximo (86), se corresponde con la última medición realizada antes de comenzar a ventilar. La diferencia es de 22 puntos.

Para ver esta información de forma numérica se inserta la Tabla 7.

MQ135 mesa

Timestamp	CO_2	Raw
12/07/2022 11:51:58	401,8555298	72
12/07/2022 11:51:58	401,8555298	72
12/07/2022 11:52:08	402,9829102	82
12/07/2022 11:52:18	403,0944824	84
12/07/2022 12:09:04	401,703949	57
12/07/2022 12:09:14	402,6874084	68
12/07/2022 12:09:24	402,8096619	69
12/07/2022 12:19:24	403,1996155	72
12/07/2022 12:29:24	403,4796753	74
12/07/2022 12:39:24	403,7006836	76
12/07/2022 12:49:25	403,9313965	77
12/07/2022 12:59:25	404,0104675	78
12/07/2022 13:09:25	404,3381042	80
12/07/2022 13:19:26	404,5958557	81
12/07/2022 13:29:26	404,8642578	81
12/07/2022 13:39:26	405,2391357	85
12/07/2022 13:49:27	405,5334167	86
12/07/2022 14:01:49	401,6684875	68
12/07/2022 14:01:59	402,1567688	80
12/07/2022 14:02:09	402,8982849	81
12/07/2022 14:12:09	401,1737061	61

Tabla 7. Segundo caso de estudio. Valores gas mesa.

En cuanto al estado de la puerta y ventana, se visualiza que se encontraban abiertas en los momentos previos a la calibración y primeras mediciones. Cuando los cambios entre cerrada-abierta-cerrada son tan bruscos, se refleja que una persona ha entrado o salido de la habitación y ha vuelto a cerrar la puerta. En el grueso de la ejecución, cuando los sensores de gas comienzan sus mediciones, ambas permanecen cerradas, hasta que, a las 13:50 se abre la ventana y comienza a ventilar la habitación y sale una persona de la misma.

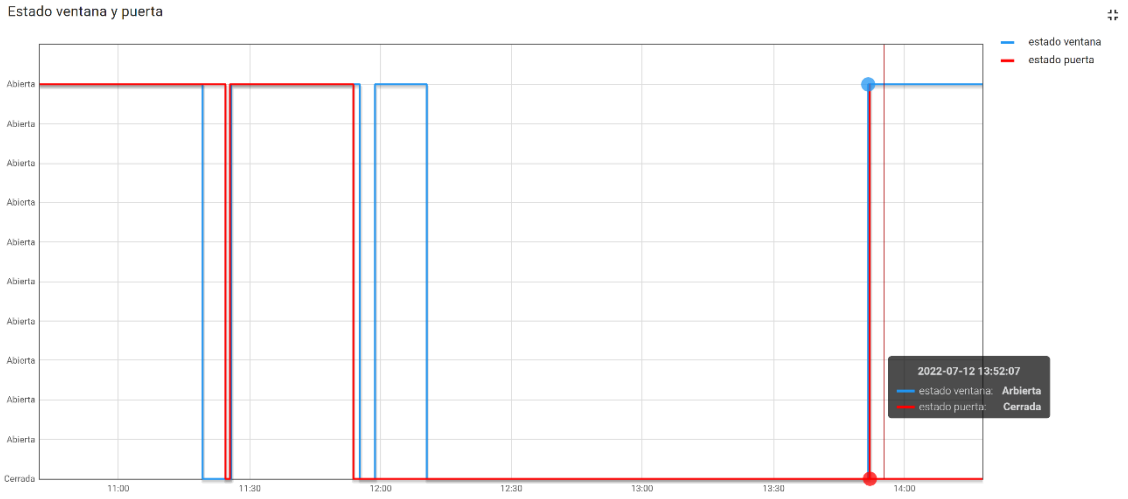


Figura 42. Segundo caso de estudio. Estado puerta y ventana.

Con respecto a la afluencia se aprecia que su funcionamiento es similar al caso de estudio anterior. Esta vez se encuentran tres personas en la habitación, aunque los valores oscilan entre 1 y 4 (al principio el 0 sí indica bien que no hay nadie en la estancia debido a la calibración de los sensores de gas). Los motivos por los que estos resultados son tan “extraños” son los mismos que los que se explicaban anteriormente en el primer caso.

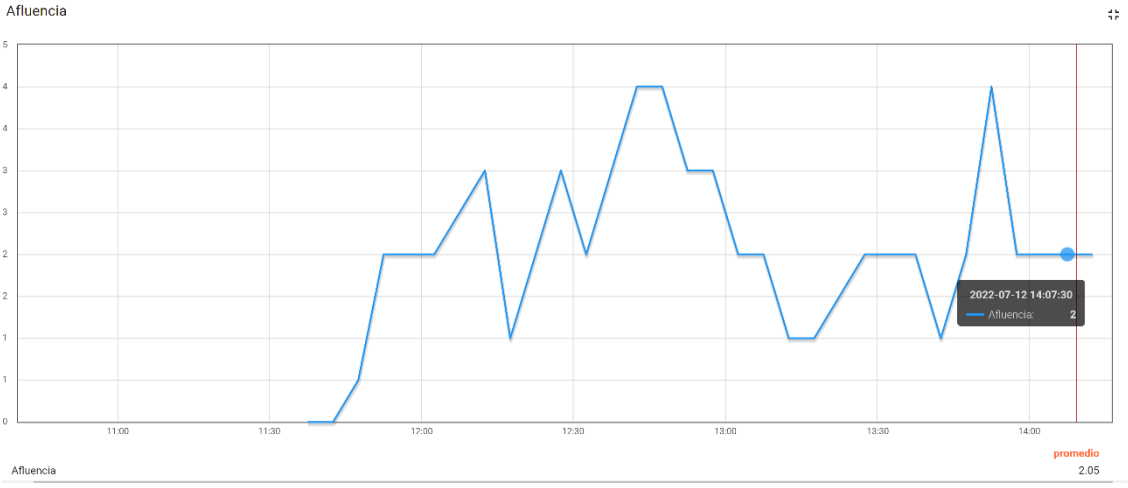


Figura 43. Segundo caso de estudio. Afluencia.

Tercer caso de estudio

En tercer lugar, se comprueba cuál es el resultado de dejar los sensores midiendo dentro de la habitación sin que haya ninguna persona en el interior.

Primero se van a repasar los valores obtenidos por el sensor ubicado cerca de la puerta.

Figura 44. Tercer caso de estudio. Gráfica gas estantería.

Al igual que en los casos anteriores, se parte de la gráfica en la que sólo se representan los valores raw. En ella, se observa que ocurre el mismo fenómeno que en el primer caso, aparece un pico máximo justo al iniciar la sesión que no se corresponde con un evento en particular. Como se ha señalado anteriormente, los sensores se calibran junto a la ventana (casi fuera) para obtener los valores más “puros” y cercanos al valor del aire limpio posibles. Por tanto, ese valor correspondería al momento en el que se introduce en la

habitación y comienza a medir. Tras ese punto, los valores obtenidos son muy cercanos entre sí, aunque, la línea presenta pequeñas oscilaciones a pesar de que no haya nadie en el interior.

Los valores oscilan entre 158 y 168, aunque, este último se podría considerar anómalo y descartarlo, ya que el siguiente valor más alto es 162, bastante más cercano a 158. En este caso, la diferencia entre los valores es bastante más baja, aun considerando el valor anómalo.

Debajo se pueden apreciar los valores numéricos que dan forma a la gráfica anterior.

MQ135 estantería

Timestamp	CO_2	Raw
13/07/2022 12:12:43	402,5780945	159
13/07/2022 12:12:53	402,9232483	161
13/07/2022 12:13:03	402,7465515	162
13/07/2022 12:23:04	403,0770264	168
13/07/2022 12:33:04	402,6333618	161
13/07/2022 12:43:04	402,5780945	159
13/07/2022 12:53:04	402,6056213	160
13/07/2022 13:03:05	402,5780945	159
13/07/2022 13:13:05	402,6056213	160
13/07/2022 13:23:05	402,5508118	159
13/07/2022 13:33:05	402,5508118	159
13/07/2022 13:43:05	402,5237427	158
13/07/2022 13:53:06	402,5237427	159
13/07/2022 14:03:11	402,4702454	158

Tabla 8. Tercer caso de estudio. Valores gas estantería.

También se revisa el comportamiento del sensor colocado próximo a la ventana.

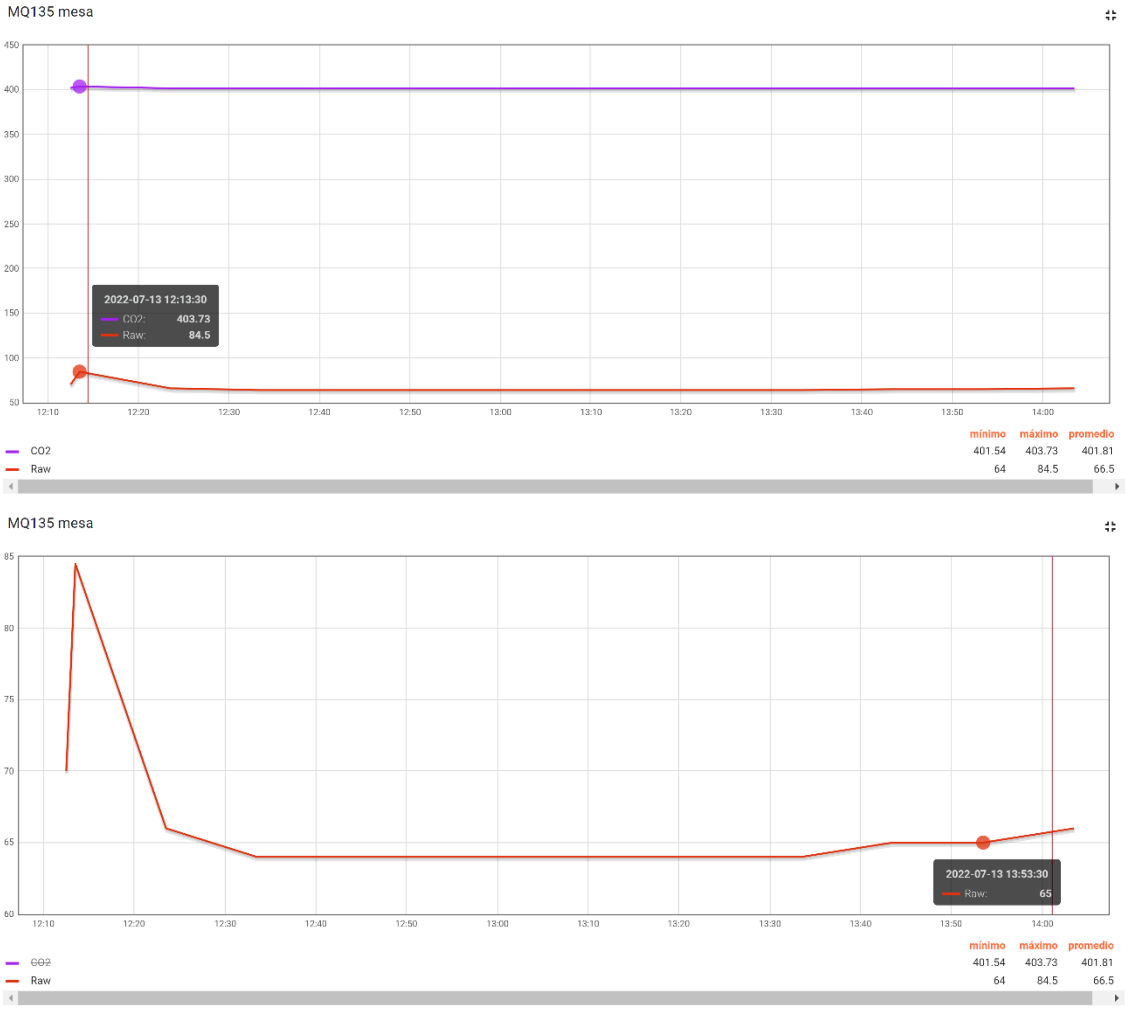


Figura 45. Tercer caso de estudio. Gráfica gas mesa.

Como se puede observar, el comportamiento de este sensor es muy similar al del sensor opuesto, a diferencia de que, en este caso, las pequeñas oscilaciones que se encuentran en el centro no se perciben y mantienen el mismo valor durante una hora.

MQ135 mesa

Timestamp	CO_2	Raw
13/07/2022 12:12:53	402,114624	70
13/07/2022 12:13:04	403,7609253	85
13/07/2022 12:13:14	403,692688	84
13/07/2022 12:23:14	401,7297668	66
13/07/2022 12:33:14	401,5385437	64
13/07/2022 12:43:14	401,5755615	64
13/07/2022 12:53:14	401,5755615	64
13/07/2022 13:03:15	401,5755615	64
13/07/2022 13:13:15	401,5755615	64

13/07/2022 13:23:15	401,5755615	64
13/07/2022 13:33:15	401,6131592	64
13/07/2022 13:43:16	401,6131592	65
13/07/2022 13:53:16	401,6131592	65
13/07/2022 14:03:16	401,6902771	66

Tabla 9. Tercer caso de estudio. Valores gas mesa.

En rango de valores va desde 64 hasta 85, aunque, tomando las consideraciones anteriores, el siguiente valor máximo sería 66, por lo que se podría considerar que, el rango es bastante menor que en el caso en el que había personas.

Revisando el estado de la puerta y la ventana, puede observarse que, en un inicio se mantienen abiertas para ventilar la habitación, en el medio de la sesión se abre y cierra la puerta en un intervalo breve (una persona entra a dejar algo en la habitación) y después se mantiene cerrado hasta el final, en el que se vuelve a abrir la ventana y la puerta.

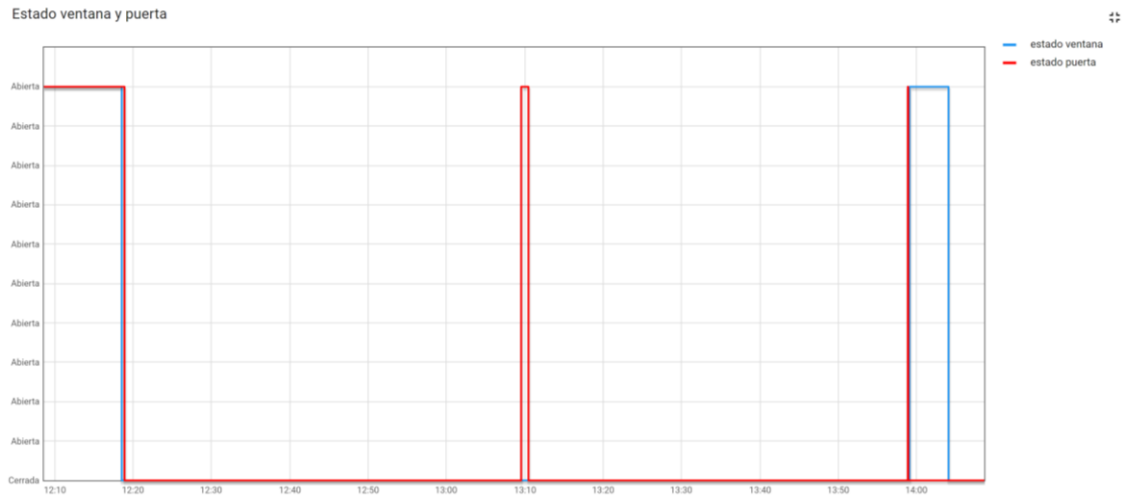


Figura 46. Tercer caso de estudio. Apertura puerta y ventana.

La afluencia en este caso sí se mantiene constante en 0, excepto al final que es cuando entro en la habitación a controlar los dispositivos. En este caso el sniffer sí ha podido captar paquetes de mi móvil.

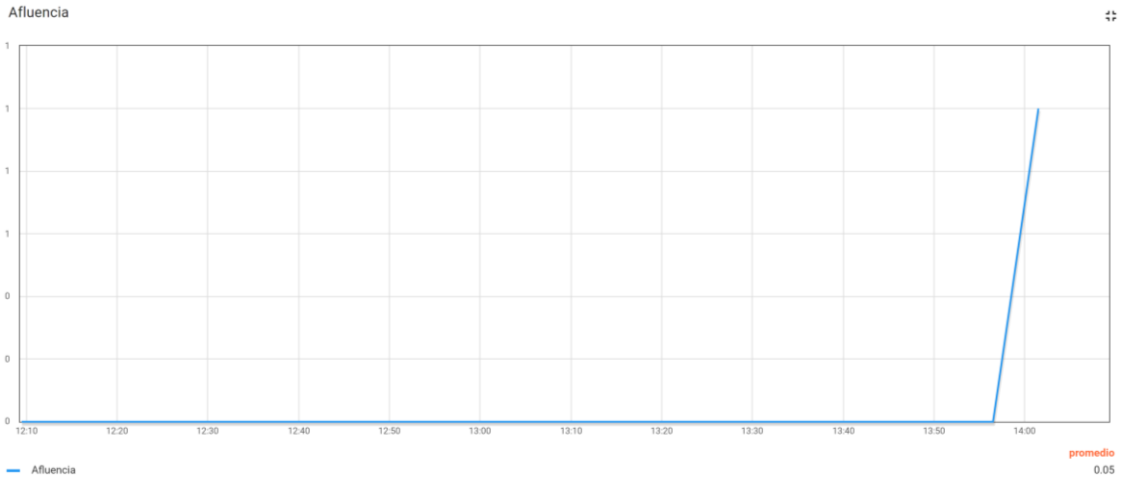


Figura 47. Tercer caso de estudio. Afluencia.

Cuarto caso de estudio

Por último, se analiza la situación en la que se encuentra la habitación con cuatro personas en una duración de hasta dos horas.

Figura 48. Cuarto caso de estudio. Gráfica gas estantería.

Comenzando con las gráficas del sensor de gas de la estantería, se ve una clara tendencia ascendente de la línea verde que representa los valores en bruto. Sin embargo, al igual que los casos anteriores, la línea que representa el CO_2 presenta cambios poco significativos con respecto a las referencias consultadas al principio del trabajo.

Pasando a la segunda gráfica en la que sólo se toman en cuenta los valores raw, puede observarse que al comienzo de la ejecución se aprecia un pequeño pico muy similar al de los casos anteriores. En este caso la causa es la misma. Tras el mínimo global (125), los valores van subiendo de forma casi constante hasta llegar al máximo (168) en el transcurso de 110 minutos. Pasado ese tiempo, se comienza a ventilar la habitación y los valores caen con mucha más

velocidad de decrecimiento, ya que, en apenas 10 minutos, pasan de 168 a 137.

El rango (valor máximo menos valor mínimo) en este caso sería de 43 puntos en 110 minutos.

En la tabla inferior pueden apreciarse los datos numéricos registrados.

MQ135 estantería

Timestamp	CO_2	Raw
13/07/2022 17:05:27	402,174469	126
13/07/2022 17:05:37	402,7580261	136
13/07/2022 17:05:47	402,5310364	128
13/07/2022 17:15:47	402,0914612	125
13/07/2022 17:25:47	402,5310364	132
13/07/2022 17:35:48	402,8255615	136
13/07/2022 17:45:48	403,4113159	144
13/07/2022 17:55:48	403,7782898	149
13/07/2022 18:05:48	403,950592	151
13/07/2022 18:15:48	404,0836792	153
13/07/2022 18:25:49	404,5032654	158
13/07/2022 18:35:49	404,8005981	160
13/07/2022 18:45:49	405,006958	162
13/07/2022 18:55:49	405,2200012	164
13/07/2022 19:05:50	405,6094055	168
13/07/2022 19:15:50	402,8943481	137
13/07/2022 19:25:50	401,6642151	116

Tabla 10. Cuarto caso de estudio. Valores gas estantería.

Pasando a las gráficas del sensor ubicado en la mesa, se aprecia a simple vista que el resultado es muy similar a su análogo.

Figura 49. Cuarto caso de estudio. Gráfica gas mesa.

Al comienzo de la ejecución se observa el pico mínimo ocasionado por los mismos motivos que el sensor de la estantería (calibrar fuera de la ventana e introducir el sensor después en la habitación). Este mínimo tiene un valor de 97. Posteriormente, los valores comienzan a subir progresivamente, aunque esta vez, en contraposición con la gráfica raw del sensor de la estantería, lo valores intermedios no difieren tanto con los anteriores, manteniendo el ritmo de crecimiento casi constante durante el grueso de la sesión. Transcurridos 110 minutos se alcanza el máximo (148), tras el cual los valores decrecen a mayor velocidad debido a la ventilación. El rango obtenido es de 51 puntos.

En la Tabla 11 se recogen los registros con los valores que se han ido midiendo.

MQ135 mesa

Timestamp	CO_2	Raw
13/07/2022 17:05:10	402,2303467	96
13/07/2022 17:05:21	403,2378845	108
13/07/2022 17:05:31	403,3344421	109
13/07/2022 17:15:31	402,3042603	97
13/07/2022 17:25:31	403,1903687	108
13/07/2022 17:35:31	403,7417297	113
13/07/2022 17:45:31	404,0700989	116
13/07/2022 17:55:32	404,359436	120
13/07/2022 18:05:32	404,9177551	123
13/07/2022 18:15:32	405,2497253	126
13/07/2022 18:25:32	405,9628601	130
13/07/2022 18:35:33	406,5029907	134
13/07/2022 18:45:33	406,9941711	137
13/07/2022 18:55:33	408,247467	144
13/07/2022 19:05:33	408,9336548	148
13/07/2022 19:15:34	403,9584045	115
13/07/2022 19:25:34	401,759613	89

Tabla 11. Cuarto caso de estudio. Valores gas mesa.

A continuación, se pasa a ver el estado de la puerta y la ventana durante la ejecución.

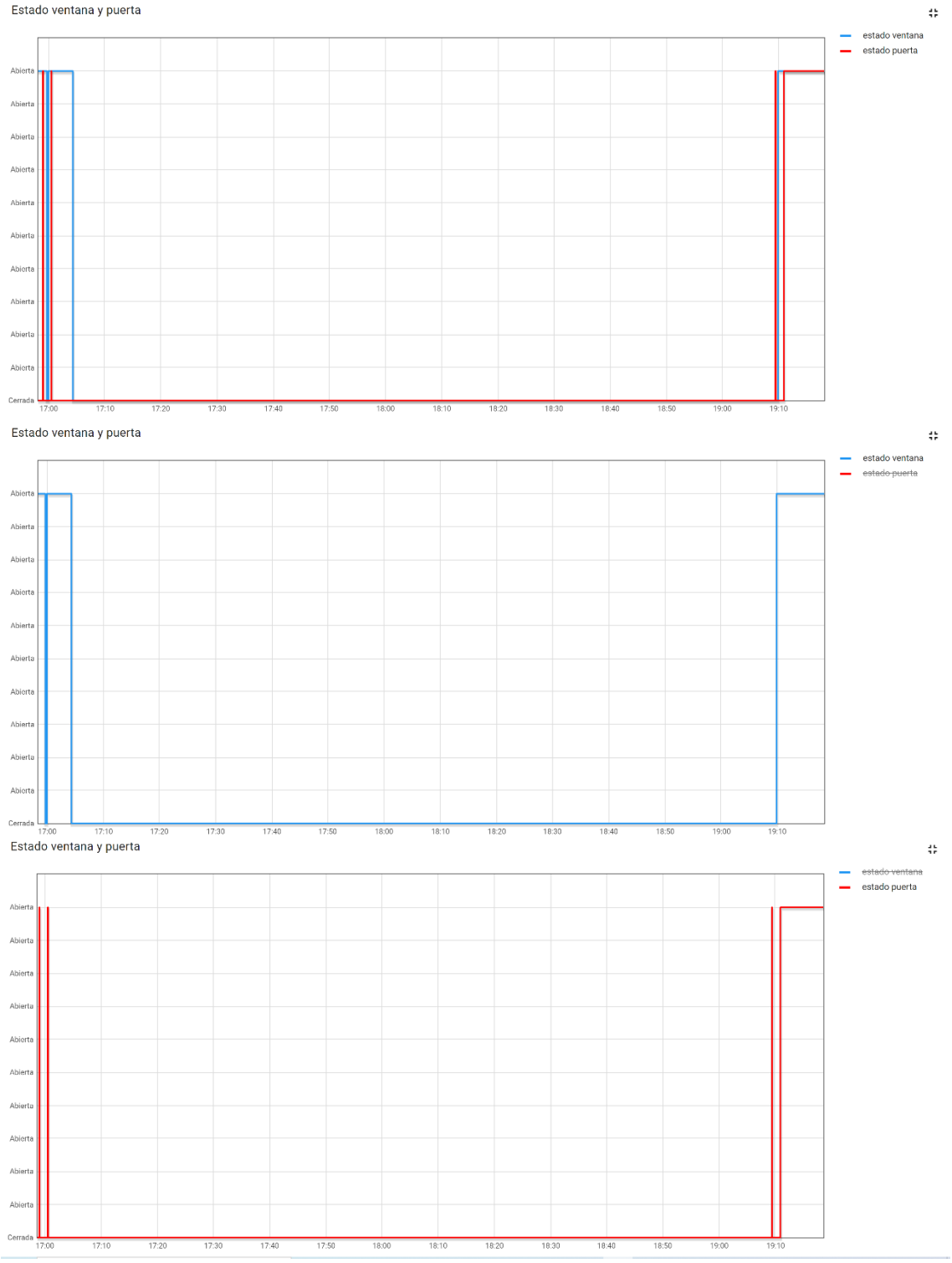


Figura 50. Cuarto caso de estudio. Apertura puerta y ventana.

.

Al comienzo de la ejecución, antes de comenzar a medir los gases, la ventana se abre para comenzar a ventilar y la puerta se mantiene cerrada excepto en

dos momentos que corresponden a una entrada y salida de la habitación. Depués, se comienzan a medir los datos de gas y la habitación se mantiene cerrada hasta el final de la misma, donde, al principio sale una persona y después se mantienen abiertas tanto la puerta como la ventana.

Por último, se procede con la gráfica de la afluencia siguiente. En ella puede apreciarse exactamente el mismo comportamiento que en las ejecuciones anteriores. A pesar de que la sesión se realiza con cuatro personas, la mayor parte del tiempo detecta dos o tres dispositivos. Incluso llega a detectar en un punto cinco. Al comienzo el valor sí es más real, ya que al principio no había nadie y luego entré yo a la habitación para comprobar que la preparación era correcta y se podía comenzar la sesión.

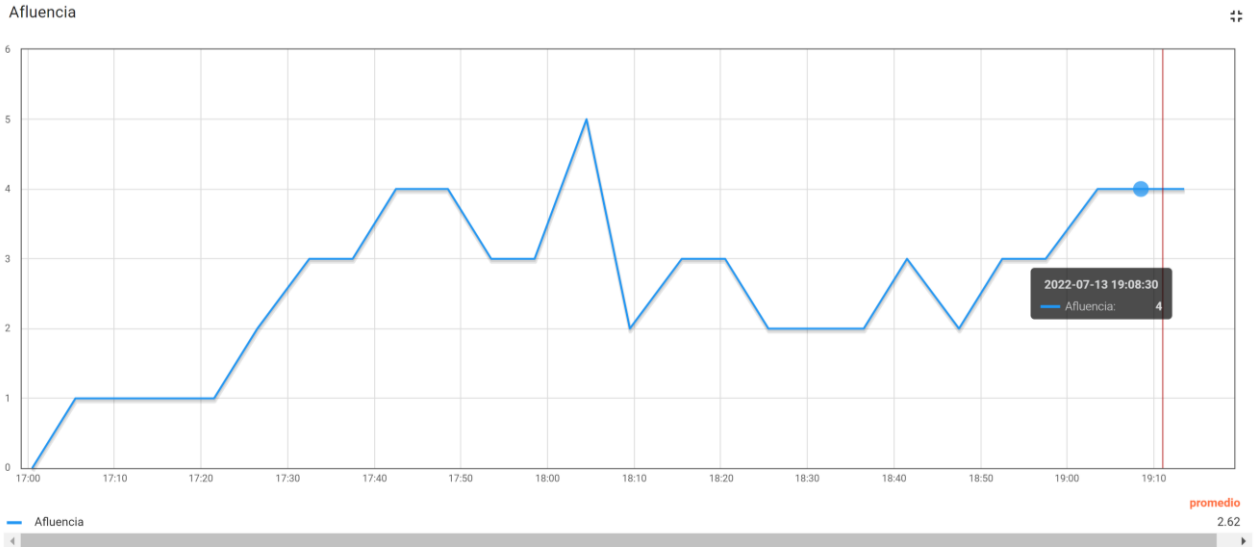


Figura 51. Cuarto caso de estudio. Afluencia.

Comentario general

Debido a que los valores que ofrece la librería en cuanto a CO_2 no son tan fidedignos con los valores que se deberían obtener, se ha decidido tomar como referencia los valores raw que obtiene el sensor. Como se conoce que los gases que es capaz de medir se encuentran NH_3, NO_x, alcohol, Benceno, humo y CO_2, y el único gas que ha podido ser alterado es el CO_2 y que puede

estar presente en ese lugar en ese momento, podemos decir que este valor en bruto del sensor representaría el CO_2 de la estancia.

Tras la realización de las diferentes experimentaciones se ha obtenido los siguientes resultados en cuanto a las mediciones de gas (en valores raw):

Número de personas	Valor máximo estantería/mesa	Valor mínimo estantería/mesa	Rango	Tiempo en alcanzar el máximo (minutos)
0	162/70	158/64	4/6	-
2	138/89	116/66	22/23	100
3	133/86	110/65	23/21	110
4	168/148	125/97	43/51	110

Tabla 12. Tabla comparativa gas en los casos de estudio.

Lo primero que llama la atención en estas ejecuciones es que cada uno de los sensores parte de un valor “cero” diferente. Mientras que el sensor de la estantería trabaja en torno a 100, el de la mesa rara vez supera esa cifra, siendo ambos sensores del mismo tipo y estando programados de la misma manera. Esto ya es un indicativo para deducir que cada sensor individual mide de forma diferente de manera intrínseca. En este punto cabe señalar que cada sensor forma parte de una placa que incorpora unas resistencias que pueden tener diferentes tolerancias, lo que puede causar estas diferencias entre unos valores y otros. Además de esto, puede haber diferencias en los materiales con los que están fabricados.

Aun así, estos valores no son erróneos, ya que el rango de ambos y el tiempo en el que se obtienen los valores máximos y mínimos siempre coinciden.

También se puede afirmar que se nota la diferencia de rango según el número de personas en las diferentes ejecuciones, ya que, a mayor número de personas, más notable será la diferencia entre el valor mínimo medido y el máximo. Como caso excepcional, puede observarse que cuando hay 2 y 3 personas los valores han sido bastante parecidos, aunque, sin embargo, al añadir una cuarta persona, el valor máximo ha alcanzado hasta 50 puntos más que el mínimo.

Por otra parte, hay que destacar que los valores de la librería no son realistas. La librería está programada para que el valor mínimo sea de 400 ppm y los

valores que ha dado en las pruebas no superaban los 410 ppm, un valor que es bastante cercano a la concentración de CO_2 en el aire libre. Aun así, las mediciones siguen los patrones de crecimiento de los valores en bruto medidos.

Además, hay que comentar que el contador de afluencia no ha funcionado como se esperaba, tal y como se ha podido comprobar a lo largo de los casos de estudio. El sniffer es capaz de detectar los paquetes que se envían mediante la tecnología Wifi, aunque, tras muchas ejecuciones y pruebas, parece que cada dispositivo tiene una frecuencia diferente en la que se envía la información y, la intensidad de estos paquetes varía bastante, aunque la posición del dispositivo permanezca estática, lo que hace que no se obtenga una información lo suficientemente consistente. Otro problema que se encuentra es la *MAC randomization*, una característica por la que los sistemas operativos intentan conseguir una mayor seguridad mediante el uso de direcciones MAC aleatorias que van cambiando para, precisamente, evitar el escaneo de dispositivos. De hecho, hay algunas fuentes que también destacan estos “problemas” [21]. Aun así, se tiene un método estimativo bastante asequible que puede proporcionar un rango de ocupantes de la estancia.

4. Conclusiones y trabajos futuros.

Este trabajo ha consistido en el diseño, implementación y experimentación de un sistema que permite observar y registrar las medidas de gas en una habitación cualquiera recogidas por dos sensores acompañadas de otra información del entorno, como es el estado de puerta y ventana y la afluencia de personas en la misma estancia.

Se puede decir que el trabajo ha resultado exitoso en cuanto a los experimentos realizados. Las concentraciones de gas aumentan progresivamente con el transcurso del tiempo y disminuyen de manera considerable cuando se abren la puerta y ventana para la ventilación.

En contraposición, la forma de contabilizar personas no ha dado los mejores resultados, debido a que los valores obtenidos no tienen la precisión que la aplicación requiere.

También cabe mencionar los problemas que ocasiona trabajar con este tipo de sensores debido a que son dispositivos que miden características analógicas del mundo y, a veces, incluso hasta la posición de un cable o las características ambientales pueden afectar a sus mediciones proporcionando más anomalías de las esperadas. Estos problemas se han encontrado al trabajar con los sensores MQ135 y MPU6050 y han afectado a los tiempos de trabajo.

Finalmente, como trabajo futuro se podrían destacar diferentes líneas:

- Análisis de los datos obtenidos por medio de modelos de ciencia de datos para obtener información más significante.
- Añadir acciones de forma proactiva. Pasando de IoT a IoE, de forma que, al alcanzar un nivel alto de gas, se pueda automatizar la apertura de puertas y ventanas para limpiar el ambiente. De forma análoga, cuando la calidad sea óptima, automatizar el cierre de puertas y ventanas para obtener un proceso eficiente en el que, en verano, no entre más calor y, en invierno, no salga más calor del necesario para obtener un ambiente limpio.

- Mejorar el mecanismo de conteo de personas mediante, por ejemplo, mecanismos de lógica difusa que determinen si un dispositivo está dentro o no, como ya se ha realizado en otros trabajos [22], o bien, utilizar otra estrategia para la contabilización, como el uso de cámaras ópticas o térmicas.

5. Apéndices.

5.1. Instalación del software.

5.1.1. ThingsBoard en Raspberry Pi model 3.

Para instalar ThingsBoard en la Raspberry, se van a seguir los pasos que se muestran en la documentación oficial.

En primer lugar, hay que instalar Java 11 mediante los comandos:

```
sudo apt update

sudo apt install openjdk-11-jdk
```

Se descarga el paquete de instalación de ThingsBoard y se instala con:

```
Sudo wget
https://github.com/thingsboard/thingsboard/releases/do
wnload/v3.3.2/thingsboard-3.3.2.deb

sudo dpkg -i thingsboard-3.3.2.deb
```

A continuación, se instala PostgreSQL:

- `wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc | sudo apt-key add -`
- `RELEASE=$(lsb_release -cs)`
- `echo "deb http://apt.postgresql.org/pub/repos/apt/ ${RELEASE}"-pgdg main | sudo tee /etc/apt/sources.list.d/pgdg.list`
- `sudo apt -y install postgresql`

Y se pone en marcha:

```
sudo service postgresql start
```

PostgreSQL crea automáticamente un usuario Unix llamado postgres que es también el superusuario por defecto de PostgreSQL. En este punto hay que cambiar a ese usuario y cambiar su contraseña en PostgreSQL mediante la consola que se inicia con el comando psql:

- `sudo su - postgres`
- `psql`
- `\password`

El siguiente paso consiste en crear la base de datos sobre la que va a trabajar ThingsBoard:

```
CREATE DATABASE thingsboard;
```

A continuación, hay que introducir la contraseña previamente creada en el archivo de configuración de ThingsBoard ubicado en /etc/thingsboard/conf/thingsboard.conf e insertarla en la línea que se muestra abajo:

```
export
SPRING_DATASOURCE_PASSWORD=YOUR_POSTGRESQL_PASSWORD_HERE
```

Después hay que ejecutar el siguiente archivo y ya se puede poner en marcha ThingsBoard.

- `sudo /usr/share/thingsboard/bin/install/install.sh`
- `sudo service thingsboard start`

En este punto se puede acceder a la interfaz de ThingsBoard introduciendo en el navegador `http://ip-equipo:8080/` teniendo en cuenta que hay que asignar al equipo una IP fija para poder "encontrarlo" siempre en la misma IP.

La configuración específica realizada en este caso se detalla en el anexo Configuración ThingsBoard. El siguiente paso sería configurar la forma en la que se va a representar la información, que aparece en el apartado Configuración dashboard y widgets.

5.1.2. ThingsBoard IoT Gateway en Raspberry Pi model 3.

Para la instalación de ThingsBoard IoT Gateway se han seguido las instrucciones de la documentación oficial. No existe una versión oficial para Raspberry, pero sí para Ubuntu, que es la que se ha utilizado.

En primer lugar, se descarga el paquete de instalación.

```
wget https://github.com/thingsboard/thingsboard-gateway/releases/latest/download/python3-thingsboard-gateway.deb
```

Seguidamente, se instala y se ejecuta.

```
sudo apt install ./python3-thingsboard-gateway.deb -y
```

Para consultar el estado, se puede ejecutar:

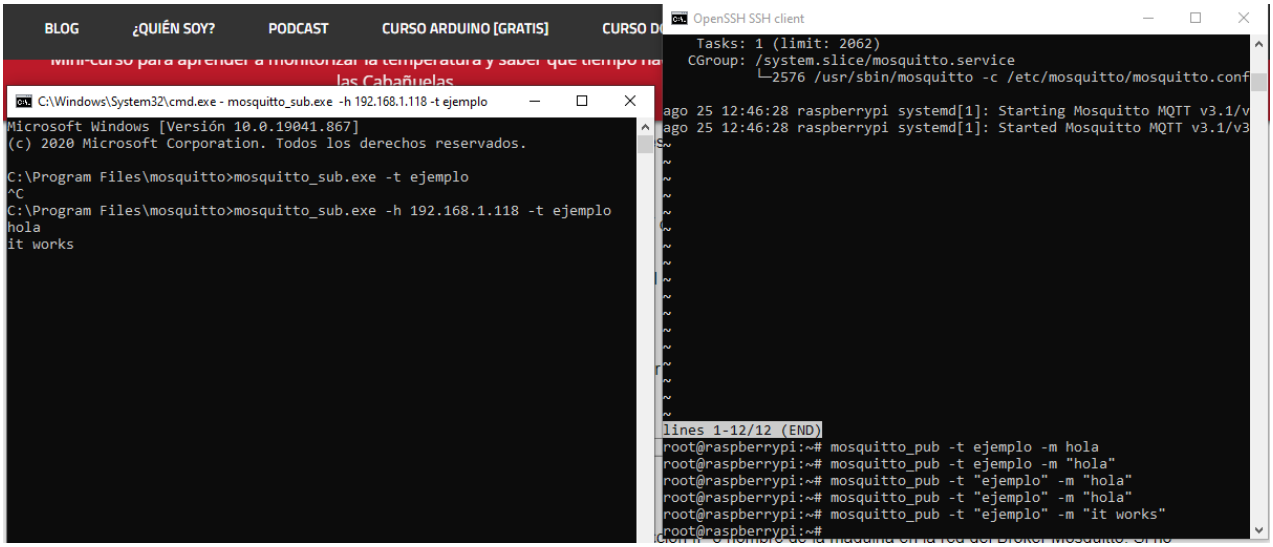
```
systemctl status thingsboard-gateway
```

El siguiente paso es realizar la configuración para mapear los mensajes del broker mosquitto a ThingsBoard mediante el conector MQTT. Este proceso se describe en el anexo Configuración ThingsBoard IoT Gateway.

5.1.3. Mosquitto en Raspberry Pi Model 3.

El siguiente paso es instalar mosquitto como broker para MQTT. Para ello se ejecuta lo siguiente:

```
sudo wget http://repo.mosquitto.org/debian/mosquitto-repo.gpg.key

sudo apt-key add mosquitto-repo.gpg.key

cd /etc/apt/sources.list.d/
```



5.2. Configuración del sistema.

5.2.1. Configuración Raspberry.

Para configurar la Raspberry es necesaria una tarjeta microSD y un ordenador auxiliar para poder instalar el sistema operativo en ella. En este caso el SO que se va a instalar es Raspberry Pi OS, como se ha indicado en apartados anteriores.

El primer paso es descargar en el ordenador el software Raspberry Pi Imager (se puede descargar aquí) e introducir la tarjeta microSD en un lector de tarjetas. El siguiente paso es ejecutar Raspberry Pi Imager (Figura 53). El proceso a seguir es bastante sencillo, se selecciona el sistema operativo que se quiere instalar (Figura 52), en nuestro caso, la primera opción y la unidad de almacenamiento que se va a utilizar. Una vez hecho esto, se pulsa Write y tras una barra de progreso habrá finalizado la instalación. Cuando finaliza aparece un mensaje indicando que el resultado es correcto y se puede extraer la tarjeta SD.

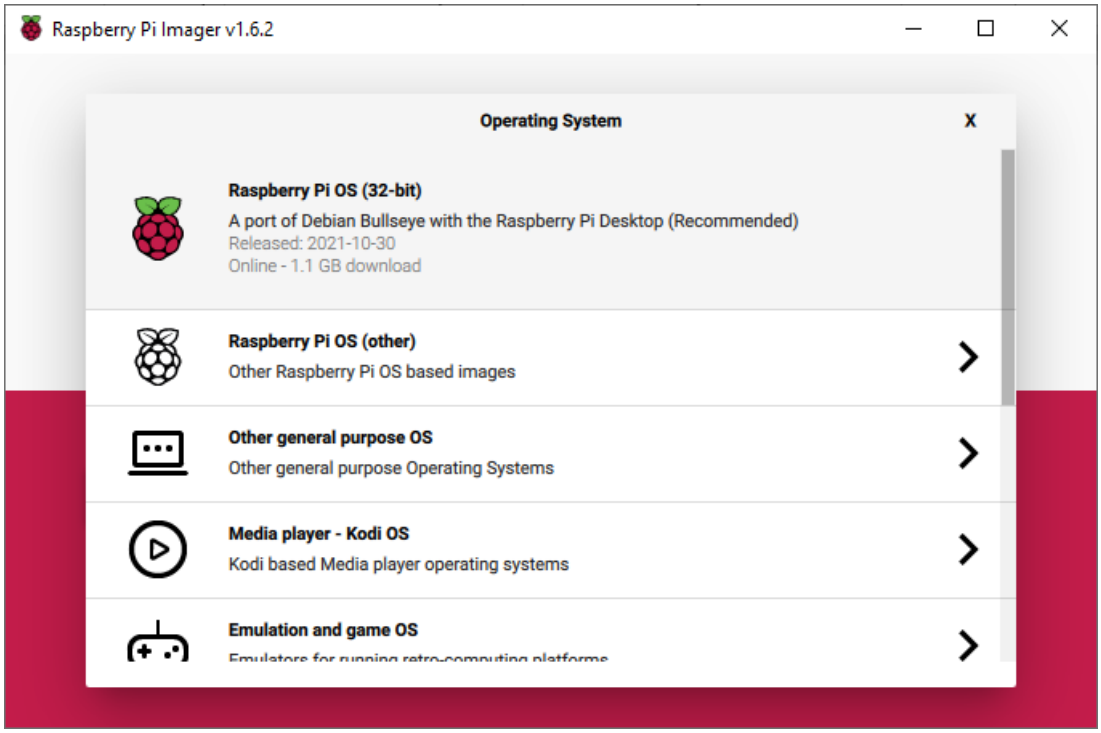


Figura 52. Captura Raspberry Pi Imager selección de sistema operativo.

Figura 53. Captura Raspberry Pi Imager.

El siguiente paso es habilitar la utilidad ssh, ya que si no se dispone de periféricos para interactuar con la Raspberry, se necesita para acceder desde otro equipo y hacer poder configurar el sistema. Para ello basta con abrir el explorador de archivos y acceder al directorio boot. Dentro de este hay que crear un archivo vacío llamado ssh. Con esto bastaría para poder habilitarla, aunque antes de ello, la Raspberry tendría que conectarse a la red para poder ejecutar ssh. Aunque se tiene disponible conexión vía Ethernet, se va a hacer uso de conexión WiFi. Para ello, se vuelve al directorio boot y se crea un archivo llamado *wpa_supplicant.conf*. El contenido del archivo debe seguir la estructura siguiente:

```
network={
ssid="nombre de la red"
psk="contraseña"
key_mgmt=WPA-PSK
}
```

Cuando se realizan estos pasos, se inserta la tarjeta microSD en la ranura de la Raspberry y se enchufa. En ese momento se encenderá un led de la placa que nos indicará que ya está en funcionamiento. En este momento hay que entrar como administrador al router (normalmente con la dirección 192.168.1.1) y, en la lista de clientes DHCP, buscar el nombre de la Raspberry y así averiguar su IP, tal y como aparecen en la Figura 54. Además de ello, se debe realizar los ajustes para que el servidor DHCP le asigne siempre la misma dirección IP y no haya que consultarla cada vez que haya que acceder mediante ssh. Para ello solo hay que introducir la dirección MAC y la IP.

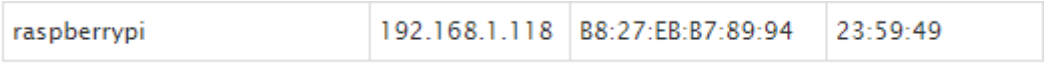


Figura 54. Captura información de red Raspberry.

En este momento ya por fin es posible acceder a la Raspberry y configurarla o realizar todas las acciones que se deseen. Simplemente escribimos en la

terminal `ssh pi@ip` e introducimos la contraseña por defecto para el usuario pi, raspberry, que después se podrá cambiar.

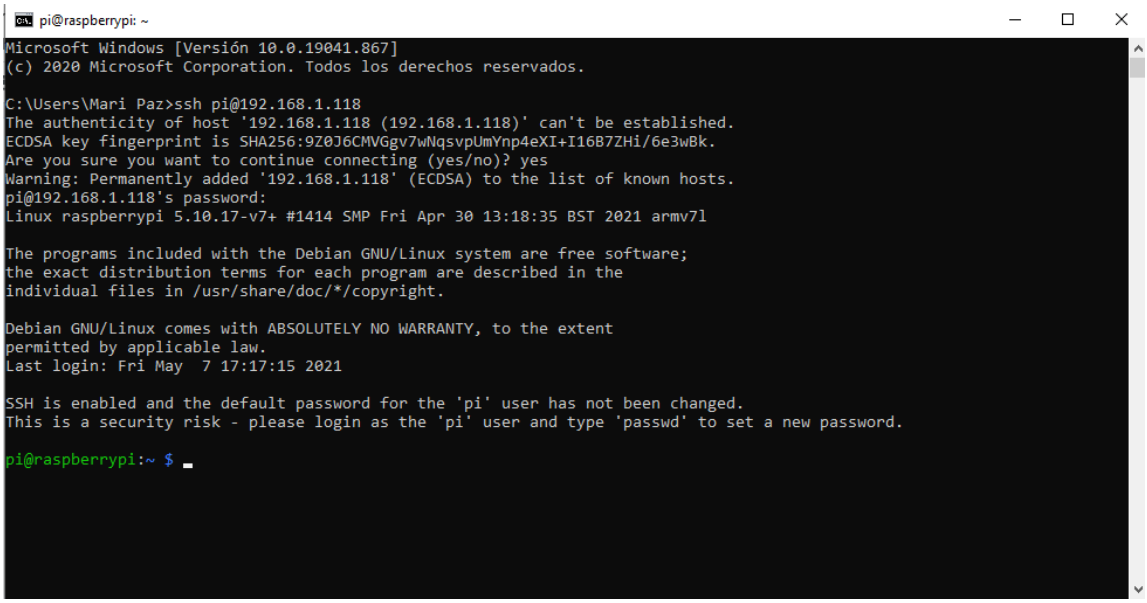


Figura 55. Conexión SSH con Raspberry.

A continuación, es recomendable ejecutar las órdenes `sudo apt update` y `sudo apt full-upgrade` para poner al día el sistema operativo.

Para la instalación de las librerías utilizadas, se ejecuta:

```
python3 -m pip install pymongo
```

5.2.2. Configuración ThingsBoard.

En primer lugar, en este caso hay que cambiar el puerto sobre el cual trabaja el broker interno de ThingsBoard, ya que por defecto es el 1883, el mismo sobre el que trabaja el broker de mosquitto. Para ello hay que acceder al archivo /etc/thingsboard/conf/thingsboard.yml y editar el campo bind-port del apartado mqtt, tal y como se muestra a continuación:

```
mqtt:
  # Enable/disable mqtt transport protocol.
  enabled: "${MQTT_ENABLED:true}"
  bind_address: "${MQTT_BIND_ADDRESS:0.0.0.0}"
  bind_port: "${MQTT_BIND_PORT:1882}"
```

Figura 56. Captura archivo de configuración thingsboard.yml.

De esta forma, los mensajes para cada broker no se van a mezclar porque van a puertos distintos.

5.2.3. Configuración dashboard y widgets.

En este apartado se va a describir el modo en el que se configuran los widgets para representar la información que obtienen los dispositivos. Esta información se representa a través de widgets colocados en un dashboard o tablero.

Antes de crear el dashboard e insertar los widgets, es necesario configurar los dispositivos, que son la fuente de los datos que se visualizarán. Para crearlos, basta con acceder al apartado gestión de dispositivos que incluye ThingsBoard (Figura 57).

Figura 57. Opción "gestión de dispositivos" de ThingsBoard.

Sin embargo, gracias a ThingsBoard Gateway, no es necesario crear los dispositivos uno por uno, ya que en los mensajes que se reciben se pueden mapear datos como el nombre del dispositivo, lo que permite que se creen automáticamente los perfiles de estos dispositivos en la plataforma. Una vez comiencen a recibirse los mensajes, se tendrá una lista de dispositivos como la que se muestra en la Figura 58.

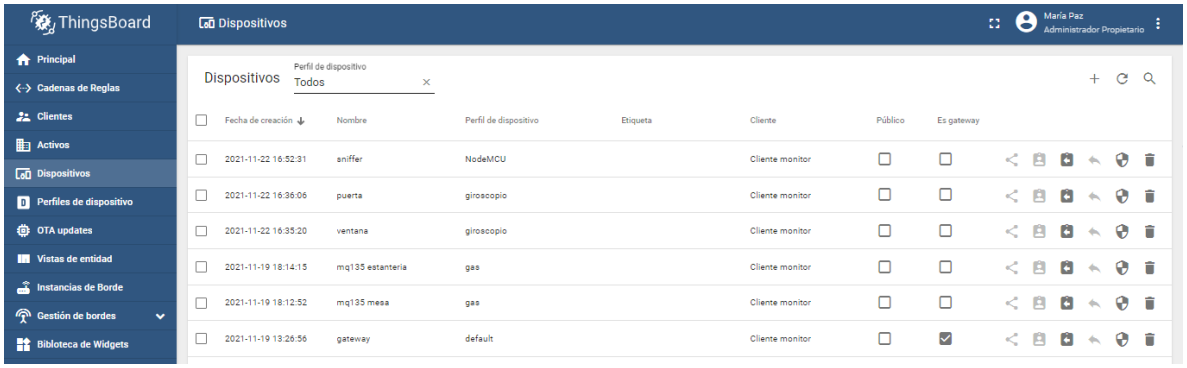


Figura 58. Apartado "dispositivos" de ThingsBoard.

Una vez se tienen los dispositivos definidos, ya se puede crear el dashboard. Para ello, se accede a la opción “paneles” de la interfaz, y se pulsa sobre el signo “+”. Se puede importar un dashboard o crearlo desde 0, que será la opción a seleccionar en primer lugar. Después de ello, se accede al panel, que tendrá una apariencia similar a la Figura 59.

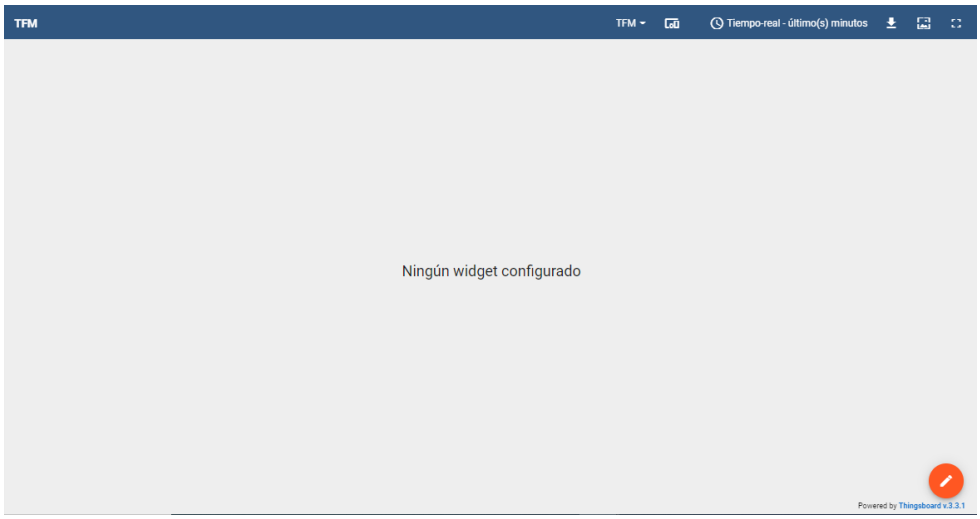


Figura 59. Captura de panel recién creado.

Para comenzar a crear widgets, simplemente hay que pulsar en el icono del lápiz de la esquina inferior derecha y entonces se podrán realizar modificaciones sobre el contenido del panel. Si se pulsa en el icono “+”, aparecen dos opciones: crear nuevo widget o importarlo. Es importante que después de cada modificación se apliquen los cambios pulsando en el tick.

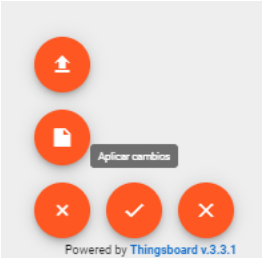


Figura 60. Botones para modificar panel e insertar widgets.

En este punto se va a explicar cómo se crea un tipo de widget específico, ya que casi todos los widgets comparten parte de la configuración principal. Se va a elegir un widget que permita representar de forma gráfica los valores del sensor MQ135. Para ello se opta por una gráfica lineal que ThingsBoard denomina Timeseries Line Chart.

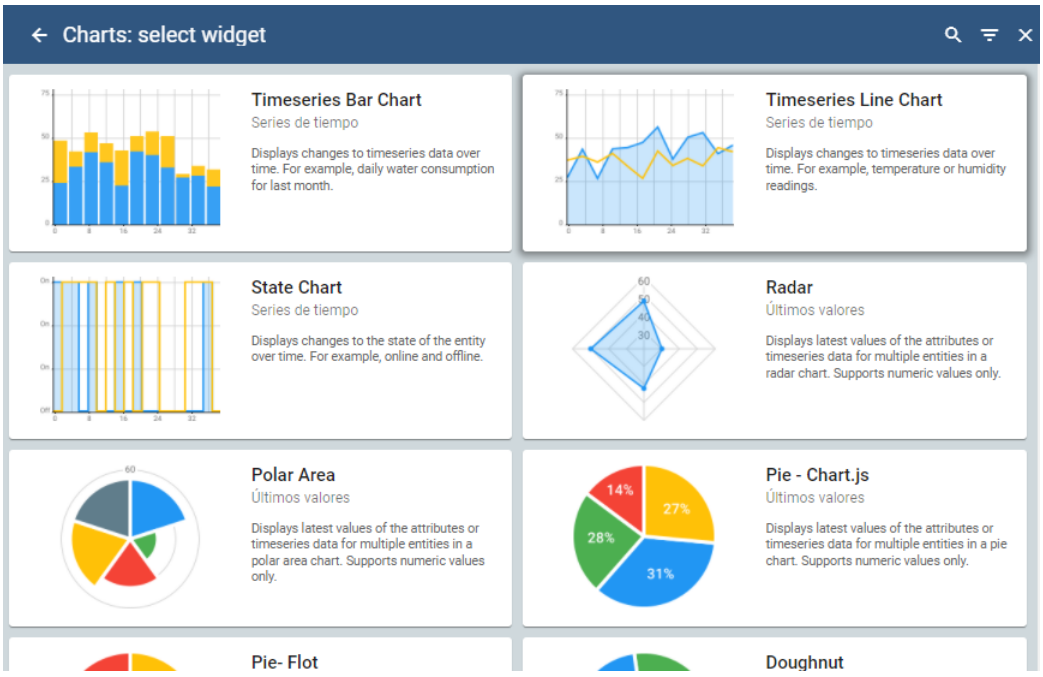


Figura 61.Algunos tipos de gráficas que incluye ThingsBoard.

Una vez seleccionado, hay que configurar la parte esencial, la fuente de los datos, desde el apartado “Set de datos”. El tipo de set puede ser tanto una entidad como una función, aunque para este caso se deja seleccionado el tipo entidad.

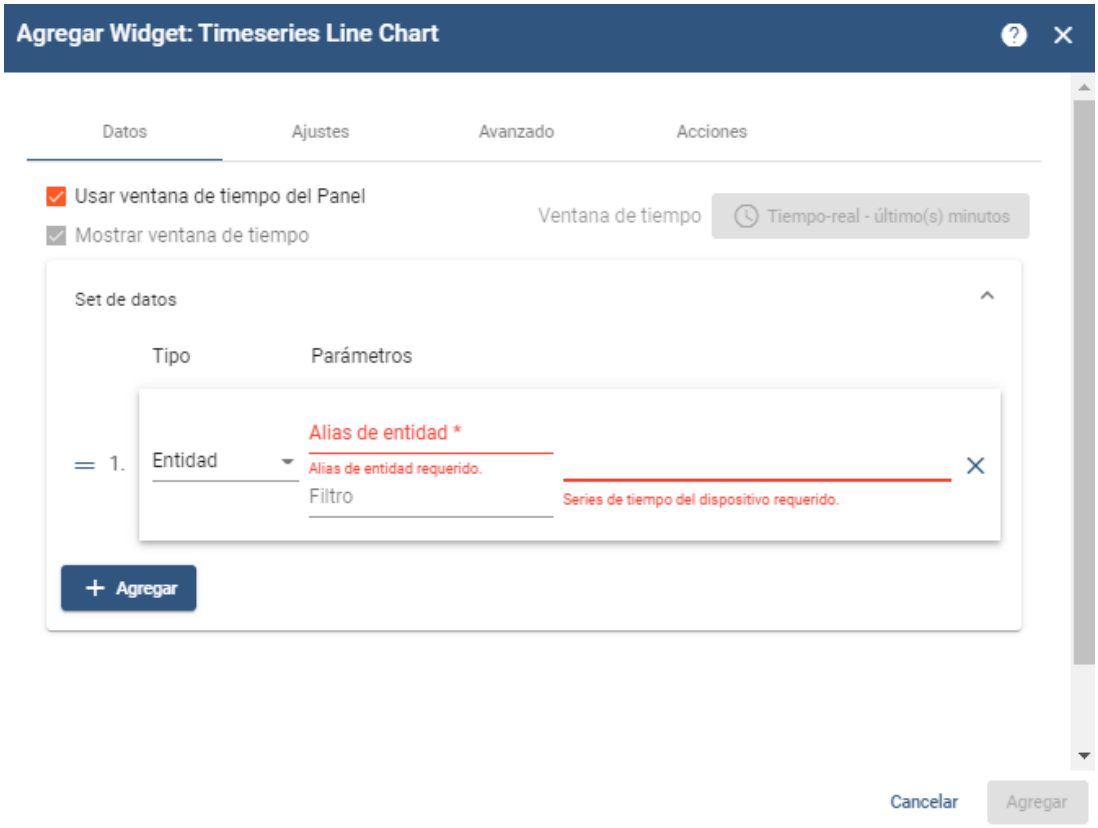


Figura 62. Configuración básica widget.

A continuación, hay que indicar un alias de entidad, que referencia a la propia fuente de los datos. Hay que añadir un nuevo alias de entidad, opción que aparece al hacer click en dicho campo. Introducimos el nombre que se quiera darle al alias (no necesariamente tiene que ser igual que el dispositivo) y se selecciona como filtro de tipo “Única entidad” y como tipo “Dispositivo”. Después, se elige el dispositivo de todos los que se encuentran registrados en la plataforma.

Figura 63. Crear nuevo alias de entidad.

Una vez se tiene la fuente de datos seleccionada, ThingsBoard permite también seleccionar las variables que representar en el widget, en el campo “timeseries” o “series de tiempo”, que se cargan automáticamente una vez introducido el dispositivo referenciado como alias. En cada caso, dependiendo del tipo de widget, se podrán añadir una o varias.

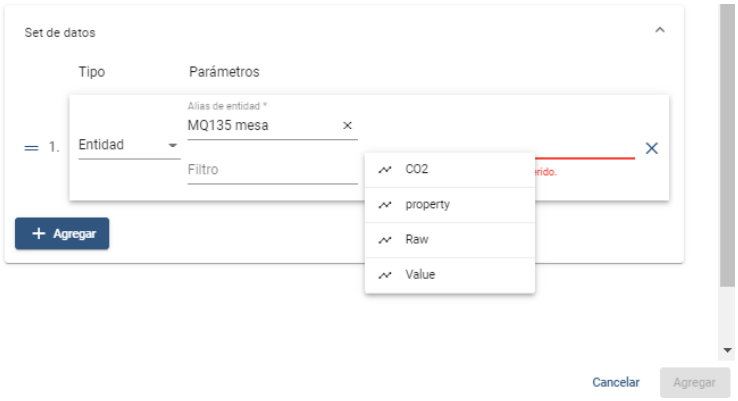


Figura 64. Variables disponibles como series de tiempo.

Se han elegido dos variables en este caso, el CO_2 y los valores Raw. Cada una de ellas se permite cambiar el color y el nombre con el que van a representarse en el widget.

Figura 65. Variables elegidas como series de tiempo.

Con esta configuración efectuada ya estaría el widget funcionando, aunque, tanto al crear el widget, como a posteriori, ThingsBoard permite realizar configuraciones más avanzadas, como añadir leyenda, dar formato a los valores mediante javascript, personalizar el estilo de los elementos…

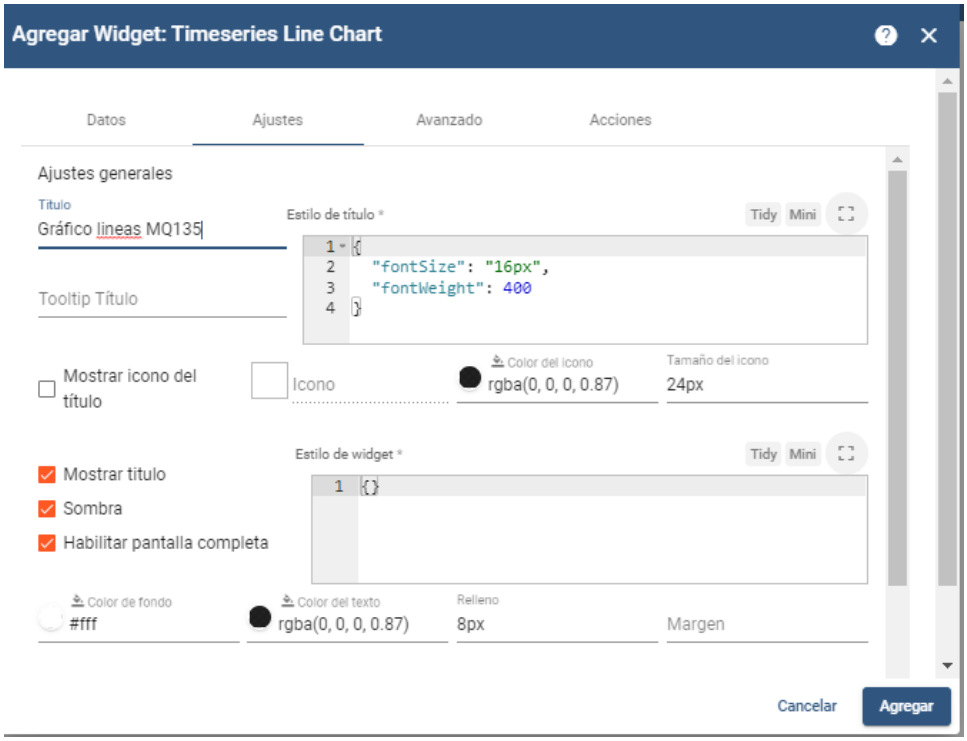


Figura 66. Ajustes adicionales al crear un nuevo widget.

Una vez realizada la configuración, se pulsa agregar y aparecerá el widget en el panel, listo para representar los datos en cuanto empiecen a llegar los mensajes de parte de los dispositivos.

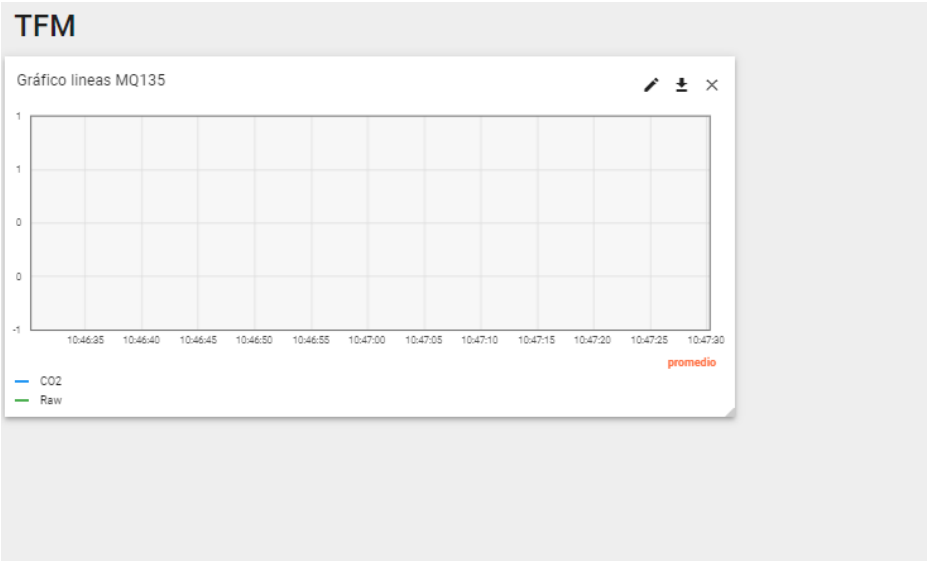


Figura 67. Widget de gráfico de líneas.

5.2.4. Configuración ThingsBoard IoT Gateway.

Para el funcionamiento de este complemento, es necesaria la modificación de dos archivos: mqtt.json y tb-gateway.yaml, ambos ubicados en /etc/thingsboard-gateway/config, en la que se ubican los ficheros de configuración de los diferentes conectores que provee (Figura 68).

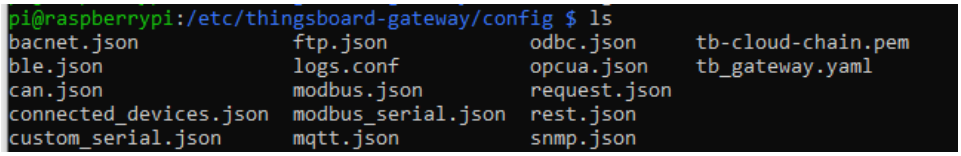


Figura 68. Archivos de configuración en /etc/thingsboard-gateway/config.

En primer lugar, es necesario crear un dispositivo e indicar que es de tipo gateway. Este dispositivo va a hacer de intermediario entre ThingsBoard y ThingsBoard Gateway. Una vez creado, hay que copiar el access token para después utilizarlo en otro fichero de configuración.

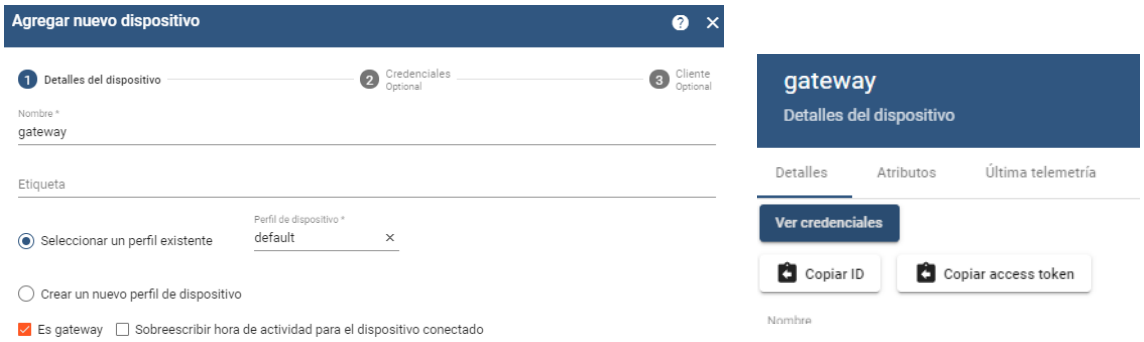


Figura 69. Configuración de dispositivo Gateway en ThingsBoard.

El siguiente paso es realizar el cambio en el archivo tb_gateway.yaml, el archivo de configuración principal. En él se indican los parámetros para la conexión del Gateway con ThingsBoard, por lo que hay que indicar el puerto y el host en el que este se encuentra. ThingsBoard y ThingsBoard Gateway se comunican mediante mqtt, por lo que hay que tener en cuenta el puerto previamente cambiado del broker interno de ThingsBoard (1882). También hay que introducir el accesToken que se ha generado anteriormente al crear el dispositivo. Además de estos cambios, hay que cerciorarse de que el conector

MQTT está sin comentar, ya que es el que se va a utilizar para comunicarse con el broker mosquitto. Una vez realizados estos cambios debería quedar como la Figura 70.

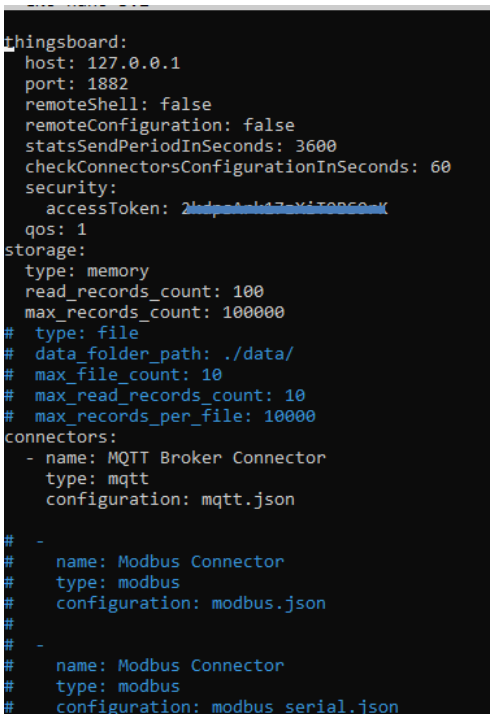


Figura 70. Captura de archivo de configuración tb_gateway.yaml.

El siguiente paso es la configuración del conector mqtt, mediante el archivo mqtt.json. Este archivo tiene diferentes partes:

- broker, donde se define la configuración del broker del que se quiere obtener los mensajes (diferente al de ThingsBoard). En este caso se encuentra en la misma máquina y en el puerto que usa MQTT por defecto, el 1883.
- mapping, la parte “principal”, en la cual se especifica un conjunto de topics a los que el Gateway se va a suscribir para recibir los mensajes y cómo los va a procesar en el apartado converter.
- connectRequest y disconnectRequest, que se utilizan en caso de querer notificar expresamente el momento de conexión y desconexión de los dispositivos. En este caso no se han realizado modificaciones en estas partes.

- attributeUpdates, que sirve para realizar el proceso inverso, obtener las modificaciones de ThingsBoard en el broker mosquitto. En este caso tampoco se va a modificar esta parte, ya que la comunicación de los dispositivos y la plataforma es unidireccional.

A continuación, se va a detallar la configuración de la sección mapping con mayor profundidad. Para cada tipo de dispositivo hay que crear una entrada en este array. En este caso al tener tres tipos diferentes (gas, giroscopio y sniffer) se han creado tres. Por ejemplo, en el caso de los sensores de gas, se van a recoger los mensajes del topic mq135/#, indicado en el campo topicFilter. En el apartado converter se especifican los campos que se van a extraer del mensaje y qué van a representar. ThingsBoard nos permite indicar el nombre de cada dispositivo y el tipo de dispositivo que es, información que se puede indicar en los campos deviceNameJsonExpression y deviceTypeJsonExpression. A estos campos se les puede asignar una variable del mensaje, en este caso deviceName y deviceType. Además de esta información estática, se pueden añadir otros atributos, como el modelo del sensor. Después se van a introducir los campos que van a indicar los valores que va midiendo el sensor en el apartado timeseries, en este caso CO2 y los datos en Raw, ambos de tipo float. Toda esta configuración se puede observar mejor en la imagen inferior.

Figura 71. Captura archivo de configuración mqtt.json.

Una vez realizados todos estos cambios, se reinicia ThingsBoard y ThingsBoard Gateway para que se aplique la configuración con `sudo systemctl restart thingsboard-gateway.service` y `sudo systemctl restart thingsboard`.

5.2.5. Configuración Aplicación.

Una vez conocida la dirección ip del broker, hay que configurar el código de los dispositivos para que sepan a qué lugar deben enviar los datos, además de insertar la información para conectarse a la red. En la variable *SSID* se introduce entre comillas el nombre de la red y en la línea inferior, la contraseña. En la variable *mqttServer* se indica el broker (puede ser una ip o un alias).

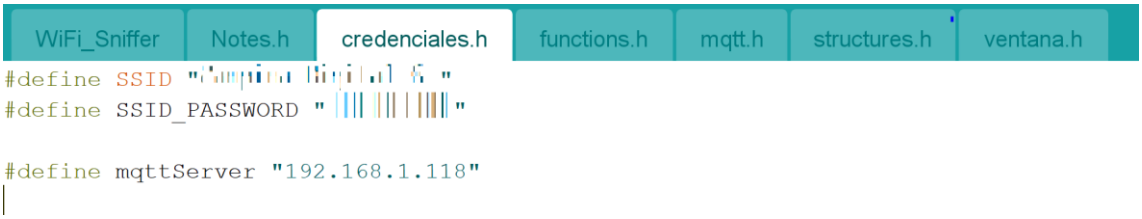


Figura 72. Archivo credenciales.h.

En el caso de querer añadir un nuevo sensor, hay que duplicar el código del tipo de sensor que se desee añadir y modificar las líneas inferiores. En este caso, para un sensor de gas, habría que cambiar la última parte del topic (estantería), con un nombre identificativo para el sensor.

```
//topics
const char* topicPublicar = "mq135/estanteria"; // mq135
const char* topicSuscripcion = "configuracion/mq135/estanteria"; // topic en el que llega el intervalo
```

Figura 73. Configuración topics.

Otro aspecto a considerar es el sniffer, ya que detectará todos los dispositivos que tengan encendida la conectividad Wifi. Por ello, lo ideal es excluir las direcciones macs que no sean representativas en cuanto a afluencia de personas se trata, por ejemplo, las de los ordenadores. Para excluir una dirección, hay que identificarla y modificar el archivo structures.h del código del sniffer. Basta con añadirla entre comillas, con las letras en minúscula y separando los pares de dígitos con :.

```
// listado direcciones MAC conocidas que no corresponden a dispositivos móviles en minúscula
std::set<String> macs_excluidas={"28:c2:dd:xx:xx:xx"};
```

Figura 74. Estructura de macs excluidas.

Por otra parte, para cambiar la frecuencia de muestreo de los sensores de gas hay que modificar el archivo de configuración ubicado en la Raspberry llamado *configuración.ini* (home/pi/Desktop/code/cliente).

En este archivo se encuentran tres apartados diferenciados: la configuración del sensor ubicado en la mesa, la configuración del sensor ubicado en la estantería y los parámetros para la conexión con la base de datos. Para cambiar la frecuencia de muestreo, simplemente hay que modificar el valor de los campos *frec_samples* y *unidad_medida*. Las líneas que comienzan con ;

hacen referencia a comentarios y el nombre de cada sección de configuración se indica entre corchetes[].

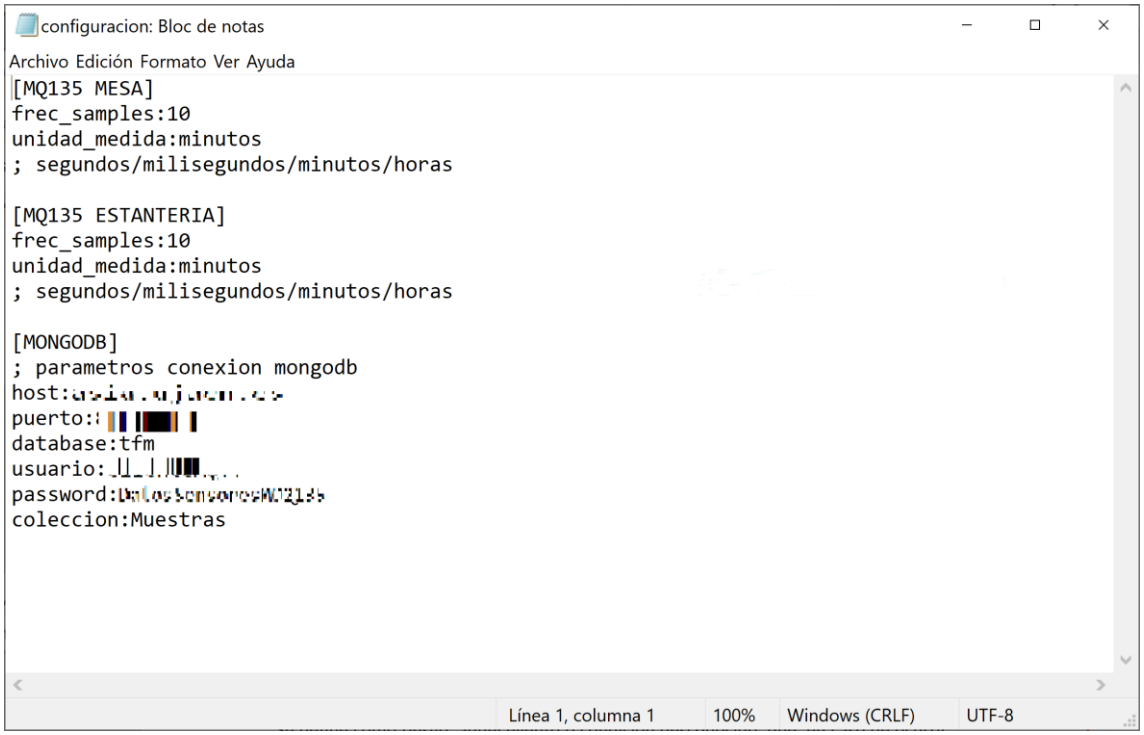


Figura 75. Captura de fichero de configuración.

En caso de querer añadir más sensores, habría que crear otro apartado de configuración entre corchetes igual a los existentes con un nombre distinto distintivo de, por ejemplo, el lugar donde se ha colocado.

5.3. Creación y gestión de bases de datos.

5.3.1. PostgreSQL.

Para eliminar los datos de los sensores que se almacenan en ThingsBoard, se pueden realizar dos tipos de consultas SQL en el terminal de PostgreSQL.

```
DELETE FROM ts_kv WHERE entity_id = 'edf87f20-495b-11ec-b1c1-f9384585c111';
```

```
thingsboard=# thingsboard=# DELETE FROM ts_kv WHERE entity_id = '33ce50e0-4bac-11ec-b1c1-f9384585c111';
DELETE 118
```

Donde entity_id se refiere al dispositivo en cuestión y se puede obtener en el menú “Dispositivos” de ThingsBoard.

Esta sentencia eliminará todos los datos de los que disponga del dispositivo, de cualquier fecha.

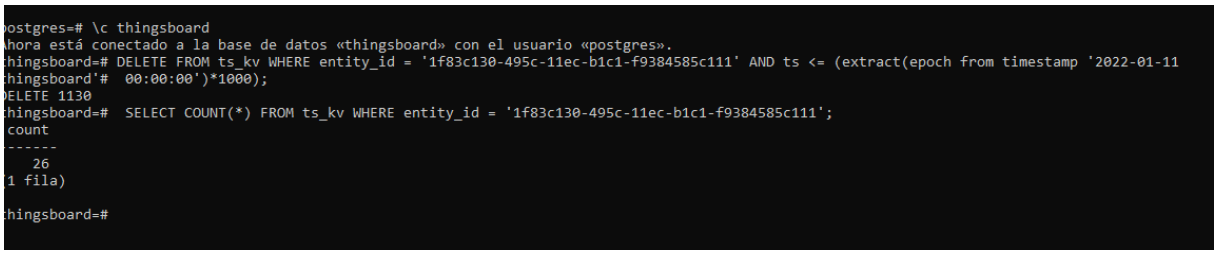
```
CREATE FUNCTION thingsboard.timeAgoMinsOrDaysOrMonths(minutes integer, days
integer, months integer)
  CALLED ON NULL INPUT
  RETURNS bigint
  LANGUAGE java AS '
    long now = System.currentTimeMillis();

    if (minutes != null)
           return new Date(now - (minutes.intValue() * 60 * 1000));
    else if (days != null)
           return new Date(now - (days.intValue() * 86400000));
    else if (months != null)
           return new Date(now - (months.intValue() * 2629746000L));
    else
          return new Date(now);

  ';
  ';
DELETE FROM ts_kv WHERE entity_id = ‘e8776de0-4ba9-11ec-b1c1-
f9384585c111’ AND ts <= extract(epoch from timestamp '2022-02-01
00:00:00');
```

e8776de0-4ba9-11ec-b1c1-f9384585c111

33ce50e0-4bac-11ec-b1c1-f9384585c111 – sniffer

1f83c130-495c-11ec-b1c1-f9384585c111 – est

DELETE FROM ts_kv WHERE entity_id = '1f83c130-495c-11ec-b1c1-f9384585c111' AND ts <= (extract(epoch from timestamp '2022-01-11 00:00:00') * 1000);



5.3.2. MongoDB.

Para la configuración de la base de datos, primero hay que acceder como admin a la misma, mediante la consola o con la utilidad gráfica Robo 3T. En este caso, se ha realizado mediante los comandos de MongoDB. Para ello se usa el comando `mongo -u admin -p contraseña servidor/admin`

```
admin
> use tfm
switched to db tfm
> use admin
switched to db admin
> show dbs
admin   0.000GB
config  0.000GB
local   0.000GB
> use tfm
switched to db tfm
> db.createUser({user: "clienteMQTT",
... pwd: passwordPrompt(),
... roles: [{role: "readWrite", db: "tfm"}],
... mechanisms: ["SCRAM-SHA-1"]
... })
Enter password:
Successfully added user: {
        "user" : "clienteMQTT",
        "roles" : [
                {
                        "role" : "readWrite",
                        "db" : "tfm"
                }
        ],
        "mechanisms" : [
                "SCRAM-SHA-1"
        ]
}
>
```

Figura 76. Creación base de datos y usuario MongoDB.

En la imagen superior se ha creado otro usuario llamado *clienteMQTT* con privilegios de lectura y escritura en la base de datos, para así no tener que acceder como administrador desde la aplicación de la Raspberry, incluso si sólo se desea consultar los últimos datos almacenados.

5.4. Manual de uso.

Para poner en marcha el sistema, el primer paso sería encender el dispositivo que va a organizar la información de todos los sensores, la Raspberry. A continuación, es necesario obtener su dirección ip y, a ser posible, fijarla para que siempre sea la misma, esta dirección nos será útil para la configuración de los dispositivos. Para una mayor comodidad, se ha configurado para que simplemente con estar conectada a la corriente se pongan en funcionamiento

todos los procesos que aloja. En este punto, se recomienda consultar el apéndice Configuración Aplicación. Por defecto, el sistema se ha configurado para que los valores de gas se midan y envíen cada 10 minutos.

Una vez encendida la Raspberry, si se accede a la dirección ip de esta, se puede acceder a ThingsBoard y comenzar a visualizar los datos.

Después, lo ideal sería comenzar a medir el estado de la puerta y la ventana, para lo cual únicamente habría que conectarlos a la alimentación, en este caso, la batería. Estos dispositivos están programados para que el estado inicial sea cerrado, por lo que deberían estar cerrados cuando se conectan a las baterías. Además de ello, es importante destacar que, para un mejor funcionamiento, cuando se abran la puerta y ventana deben alcanzar un ángulo cercano a 90º.

El siguiente paso puede ser comenzar a controlar la afluencia de la habitación. Para ello, basta con conectar la placa en la que se encuentre el sniffer a la corriente. Para que recoja los dispositivos de forma efectiva, es necesario colocarlo en el centro de la estancia.

Para poner en marcha el sistema en su totalidad es necesaria una pequeña preparación previa, ya que los sensores necesitan alcanzar una determinada temperatura para que las mediciones sean lo más correctas posible.

En este paso es muy importante que el sensor se calibre a una determinada temperatura y en un ambiente considerado “limpio”. Por estos motivos, la placa ha sido programada para que caliente los 10 primeros minutos tras ser conectada a la corriente. De esta forma alcanzará una temperatura adecuada para, posteriormente, tomar el valor base que considerará de aire limpio, con baja concentración de CO_2. El usuario se limitará a enchufar la placa que contenga el sensor y tratar de ventilar la habitación lo máximo posible (abrir las ventanas y puertas). Es aconsejable que este paso se intente realizar con el menor número de personas en la estancia posible, ya que los sensores deben medir el aire con la mayor calidad posible.

6. Definiciones y abreviaturas.

En este apartado se muestran las definiciones de palabras clave para una mejor comprensión del trabajo.

-Internet de las cosas (IoT): Es la red de objetos físicos que incluyen tecnología incorporada para comunicarse y percibir o interactuar con sus estados internos o el entorno externo [6].

-Computación ubicua: definida por Friedemann Mattern como "omnipresencia de computadores muy pequeños interconectados sin cables que se incrustan de forma casi invisible en cualquier tipo de objeto cotidiano." Algunas características principales de la computación ubicua son [23]:

- Compuesto por computadores conectados entre sí, distribuidos y accesibles de forma transparente.
- La interacción hombre–máquina (HCI) es más natural, invisible.
- Los computadores tienen consciencia del contexto.
- Puede existir trabajo autónomo por parte de los computadores.
- Podría tomar decisiones de manera inteligente (IA).

-Sensor: dispositivo capaz de captar magnitudes de su entorno.

-Ley de Ohm: Esta ley establece que la relación entre voltaje (V), corriente (I) y resistencia (R) de un circuito eléctrico viene determinada por la expresión

$$V = R * I$$

-Divisor de tensión: Es un circuito electrónico que permite dividir la tensión (o voltaje) de entrada en dos más pequeñas de salida. De esta forma se puede alimentar un componente con un voltaje inferior al que proporciona una fuente de alimentación. Se basa en la Ley de Ohm, y su expresión viene dada por:

$$V_{out} = \frac{R_2}{R_1 + R_2} * V_{in}$$

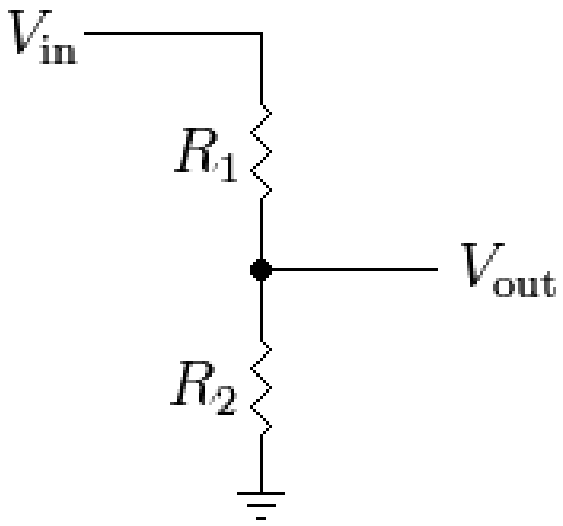


Figura 77. Divisor de tensión resistivo, Wikipedia [24].

-Pull down: tipo de circuito que permite asegurar un valor lógico bien definido (alto o bajo). En este caso, cuando el interruptor se cierra, el valor que se obtiene es alto, y cuando está abierto permanece a nivel bajo. Se coloca una resistencia entre la salida y la tierra, tal y como se muestra en la Figura 78.

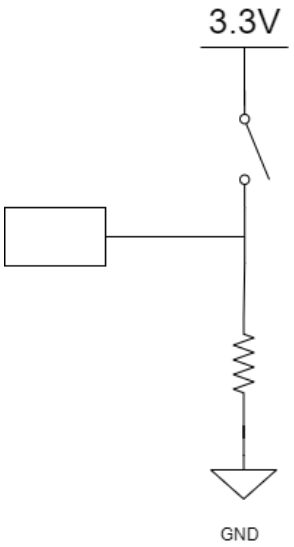


Figura 78. Circuito con resistencia pull down.

-Plataforma IoT: Una plataforma IoT es un conjunto de software local o un servicio en la nube (plataforma IoT como servicio [PaaS]) que supervisa y puede gestionar y controlar varios tipos de dispositivos, a menudo a través de aplicaciones que las unidades de negocio despliegan en la plataforma [25].

-Acelerómetro: Es un sensor capaz de medir la aceleración de un objeto, los cambios en su velocidad. Pueden detectar desde la fuerza de la gravedad hasta pequeñas vibraciones. Estas mediciones se expresan en m/s^2 o en fuerzas G (g) [26].

En este caso se ha usado el módulo HW 616, que incorpora el acelerómetro MMA 8452 (Figura 79), de tres ejes, 8/12 bits de resolución y bajo consumo. Este dispositivo tiene un rango de 2 g/ 4g / 8g, lo que permite adaptarlo según el tipo de movimiento (más brusco o más leve). Para las comunicaciones con el microcontrolador hacen uso de la interfaz I^2C mediante los pines SCL y SDA. Suelen usarse, por ejemplo, en teléfonos móviles para obtener la orientación de estos, o para detectar movimiento o caídas de otros dispositivos [27].

Figura 79. Acelerómetro HW 616.

-Giroscopio: Es un sensor capaz de medir la velocidad angular, los cambios en el ángulo rotacional del dispositivo por unidad de tiempo. Esta magnitud se expresa en grados por segundo.

Para el trabajo se ha utilizado el módulo MPU 6050 (Figura 80), que consta de un acelerómetro y un giroscopio, ambos de tres ejes, una resolución de 16 bits y bajo consumo. Además de esto, incorpora un DMP (Digital Motion Processor) para procesar las señales. Tanto el giroscopio como el acelerómetro de este módulo se puede adaptar según el tipo de movimiento (±250/±500/±1000/±2000°/seg (dps) para giroscopio y ±2g/±4g/±8g/ ±16g para

el acelerómetro). Para las comunicaciones con el microcontrolador hacen uso de la interfaz I^2C mediante los pines SCL y SDA. Las principales aplicaciones de este módulo se encuentran en teléfonos y tablets, sobre todo relacionadas con la salud y estado físico [28].

Figura 80. Módulo MPU 6050.

-Sensor MQ135: Sensor de gas de tipo metal óxido semiconductor utilizado para controlar la calidad del aire capaz de detectar diferentes gases, como NH_3, NO_x, alcohol, Benceno, humo, CO_2 … [29]

Su funcionamiento se basa en las variaciones de su resistencia al contactar con el gas, previamente el sensor se encuentre en su temperatura óptima de trabajo. Este proceso es crucial, por lo que el sensor incluye un elemento calentador para alcanzar esta temperatura y que así los materiales de los que está compuesto adquieran la sensibilidad necesaria.

El sensor en sí mismo se encuentra bajo la cúpula plateada con membrana que se aprecia en la Figura 81, aunque normalmente se comercializan en módulos (la placa azul) que incluyen a su vez otros componentes.

Figura 81. Módulo con sensor MQ135.

-Sniffer: Es una herramienta que puede ser tanto software como hardware que permite monitorizar el tráfico de red en tiempo real. Funcionan capturando y analizando los paquetes que circulan por la red [30].

-NoSQL: Categoría general de Sistemas Gestores de Bases de Datos que se caracteriza por:

- No utilizar SQL como lenguaje de consultas.
- No utilizar estructuras fijas (tablas-relaciones).
- No suelen permitir operaciones de tipo Join.
- Almacenamiento desnormalizado.
- Arquitectura distribuida.

Este tipo de SGBD permite ejecutarse en máquinas con pocos recursos, es adecuado para manejar un gran número de datos gracias a las estructuras que utiliza y es escalable de forma horizontal [31]. MongoDB es un ejemplo de SGBD NoSQL.

7. Bibliografía.

[1] C. C. Wang *et al.*, «Airborne transmission of respiratory viruses», *Science*, ago. 2021, doi: 10.1126/science.abd9149.

[2] Ministerio de Sanidad. Gobierno de España, «Ventilación y COVID-19». [En línea]. Disponible en: https://www.sanidad.gob.es/profesionales/saludPublica/ccayes/alertasActual/nCov/documentos/COVID19_Ventilacion.pdf

[3] «El Gobierno publica recomendaciones sobre el uso de sistemas de climatización y ventilación para prevenir la expansión del COVID-19». https://www.miteco.gob.es/es/ministerio/medidas-covid19/sistemas-climatizacion-ventilacion/default.aspx

[4] U. Satish *et al.*, «Is CO2 an indoor pollutant? direct effects of low-to-moderate CO2 concentrations on human decision-making performance», *Environ. Health Perspect.*, vol. 120, n.º 12, pp. 1671-1677, 2012, doi: 10.1289/ehp.1104789.

[5] F. Khodadadi, A. V. Dastjerdi, y R. Buyya, «Chapter 1 - Internet of Things: an overview», en *Internet of Things*, R. Buyya y A. Vahid Dastjerdi, Eds. Morgan Kaufmann, 2016, pp. 3-27. doi: 10.1016/B978-0-12-805395-9.00001-0.

[6] «Definition of Internet Of Things (iot) - Gartner Information Technology Glossary», *Gartner*. https://www.gartner.com/en/information-technology/glossary/internet-of-things.

[7] D. Evans, «How the Next Evolution of the Internet Is Changing Everything», p. 11, 2011.

[8] «The Internet of Everything - Global Public Sector Economic Analysis». 2013.

[9] D. Korzun, E. Balandina, A. Kashevnik, S. Balandin, y F. Viola, *Ambient Intelligence Services in IoT Environments: Emerging Research and Opportunities*. IGI Global, 1d. C. [En línea]. Disponible en: https://www.igi-global.com/book/ambient-

intelligence-services-iot-environments/www.igi-global.com/book/ambient-intelligence-services-iot-environments/218560

[10] «UJAmI SmartLab | UJAmI». https://ceatic.ujaen.es/ujami/en/smartlab

[11] Z. Yunusa, M. N. Hamidon, A. Kaiser, y Z. Awang, «Gas sensors: A review», *Sens. Transducers*, vol. 168, n.º 4, pp. 61-75, 2014.

[12] A. Agrawal, «Agile methodology: Incremental and Iterative way of development», *Medium*, 5 de diciembre de 2019. https://medium.com/@ashutoshagrawal1010/agile-methodology-incremental-and-iterative-way-of-development-a6614116ae68

[13] «MQTT and CoAP, IoT Protocols | The Eclipse Foundation». https://www.eclipse.org/community/eclipse_newsletter/2014/february/article2.php

[14] «What is MQTT? Why use MQTT? Why MQTT is one of the best network protocols for the Internet of Things», *IBM Developer*, 9 de diciembre de 2021. https://developer.ibm.com/articles/iot-mqtt-why-good-for-iot/

[15] «NodeMCU, la popular placa de desarrollo con ESP8266», *Luis Llamas*. https://www.luisllamas.es/esp8266-nodemcu/

[16] «What is a Raspberry Pi?», *Raspberry Pi*. https://www.raspberrypi.org/help/what-is-a-raspberry-pi/

[17] «¿Qué es MongoDB?», *MongoDB*. https://www.mongodb.com/es/what-is-mongodb

[18] Thingsboard, «ThingsBoard installation options», *ThingsBoard*. https://thingsboard.io/docs/user-guide/install/installation-options/

[19] Thingsboard, «What is ThingsBoard?», *ThingsBoard*. https://thingsboard.io/docs/getting-started-guides/what-is-thingsboard/ (accedido

[20] SensorsIOT, *Wi-Fi Sniffer as a Human detector*. 2022. [En línea]. Disponible en: https://github.com/SensorsIot/Wi-Fi-Sniffer-as-a-Human-detector

[21] A. Farah, «7 Technologies that Count People (Buildings, Offices & Agnostic)», *Density*, 9 de octubre de 2020. https://medium.com/density-inc/7-technologies-that-count-people-buildings-offices-742785d2030f

[22] A.-P. Albín-Rodríguez, Y.-M. De-La-Fuente-Robles, J.-L. López-Ruiz, Á. Verdejo-Espinosa, y M. Espinilla Estévez, «UJAmI Location: A Fuzzy Indoor Location System for the Elderly», *Int. J. Environ. Res. Public. Health*, vol. 18, n.º 16, p. 8326, ago. 2021, doi: 10.3390/ijerph18168326.

[23] «Módulo 1 - Computación ubicua. Tema 1 - Introducción a la Computación Ubicua. Apuntes de la asignatura Sistemas Empotrados y Ubicuos.»

[24] *Resistive divider - Wikipedia*. [En línea]. Disponible en: https://en.wikipedia.org/wiki/File:Resistive_divider.png

[25] «Definition of IoT Platforms - Gartner Information Technology Glossary», *Gartner*. https://www.gartner.com/en/information-technology/glossary/iot-platforms

[26] «ABC del acelerometro». https://www.5hertz.com/index.php?route=tutoriales/tutorial&tutorial_id=2

[27] NXP Semiconductors, «MMA8452Q 3-axis, 12-bit/8-bit digital accelerometer datasheet». 10 de abril de 2016.

[28] InvenSense, «MPU-6000 and MPU-6050 Product Specification Revision 3.4». 19 de agosto de 2013.

[29] Olimex, «Technical data MQ-135 gas sensor».

[30] «What Is a Sniffer and How Can You Prevent Sniffing?», *What Is a Sniffer and How Can You Prevent Sniffing?* https://www.avg.com/en/signal/what-is-sniffer

[31] «Tema 0 - Bases de datos NoSQL, Introducción. Apuntes de la asignatura Computación distribuida para la gestión de datos a gran escala.»