



UNIVERSIDAD DE JAÉN  
*Escuela Politécnica Superior de Jaén*

Trabajo Fin de Grado

# SISTEMA IoT PARA LA MONITORIZACIÓN DE GASES Y TEMPERATURA DE DEPOSICIONES

**Alumno: David Díaz Jiménez**

Tutor: Macarena Espinilla Estévez  
José Luis López Ruiz

Dpto: Informática

**Septiembre, 2021**



# Índice

|                                                |           |
|------------------------------------------------|-----------|
| <b>Índice de figuras</b>                       | <b>4</b>  |
| <b>Índice de tablas</b>                        | <b>4</b>  |
| <b>1. Introducción</b>                         | <b>9</b>  |
| 1.1. Contextualización del proyecto . . . . .  | 9         |
| 1.2. Propósito general y objetivos . . . . .   | 11        |
| 1.2.1. Objetivo general . . . . .              | 11        |
| 1.2.2. Objetivos específicos . . . . .         | 11        |
| 1.3. Revisión de trabajos existentes . . . . . | 12        |
| 1.4. Resultados esperados . . . . .            | 13        |
| 1.5. Estructura de la memoria . . . . .        | 13        |
| <b>2. Análisis</b>                             | <b>15</b> |
| 2.1. Problemas a abordar . . . . .             | 15        |
| 2.2. Análisis de requisitos . . . . .          | 15        |
| 2.2.1. Requisitos funcionales . . . . .        | 15        |
| 2.2.2. Requisitos no funcionales . . . . .     | 17        |
| <b>3. Planificación</b>                        | <b>20</b> |
| 3.1. Metodología . . . . .                     | 20        |
| 3.1.1. Principales metodologías . . . . .      | 20        |
| 3.1.2. Elección de la metodología . . . . .    | 25        |
| 3.2. Planificación de tareas . . . . .         | 26        |
| 3.3. Diagrama de Gantt . . . . .               | 28        |
| 3.4. Estimación de costes . . . . .            | 30        |
| 3.4.1. Costes humanos . . . . .                | 31        |
| 3.4.2. Costes de hardware y software . . . . . | 37        |
| 3.4.3. Costes adicionales . . . . .            | 39        |
| 3.4.4. Costes totales . . . . .                | 40        |
| <b>4. Diseño</b>                               | <b>42</b> |
| 4.1. Arquitectura del sistema IoT . . . . .    | 42        |

|                                                                         |           |
|-------------------------------------------------------------------------|-----------|
| 4.2. Diseño software . . . . .                                          | 44        |
| 4.2.1. Diagrama de flujo . . . . .                                      | 44        |
| 4.2.2. Casos de uso . . . . .                                           | 45        |
| 4.3. Diseño de la interacción entre todo el sistema . . . . .           | 53        |
| 4.4. Diseño hardware del dispositivo . . . . .                          | 59        |
| 4.4.1. Placas de desarrollo . . . . .                                   | 59        |
| 4.4.2. Sensores . . . . .                                               | 64        |
| 4.4.3. Actuadores . . . . .                                             | 71        |
| 4.4.4. Iteraciones realizadas . . . . .                                 | 71        |
| 4.5. Diseño físico del dispositivo . . . . .                            | 74        |
| 4.5.1. Iteraciones del modelo 3D . . . . .                              | 75        |
| 4.6. Diseño del modelo de datos . . . . .                               | 79        |
| 4.6.1. Entidades del modelo de datos . . . . .                          | 80        |
| 4.6.2. Justificación de la elección de MongoDB . . . . .                | 80        |
| 4.6.3. Colecciones de datos . . . . .                                   | 83        |
| <b>5. Implementación</b>                                                | <b>88</b> |
| 5.1. Entorno de desarrollo . . . . .                                    | 88        |
| 5.2. Sumidero de datos . . . . .                                        | 91        |
| 5.2.1. MQTT . . . . .                                                   | 91        |
| 5.2.2. Transformación de los datos de temperatura en imágenes . . . . . | 97        |
| 5.3. Sistema para la recogida de datos . . . . .                        | 99        |
| 5.3.1. Sistema maestro-esclavo . . . . .                                | 100       |
| 5.3.2. Transmisión de datos . . . . .                                   | 101       |
| 5.3.3. MQTT . . . . .                                                   | 102       |
| 5.3.4. Serialización de los datos . . . . .                             | 104       |
| 5.3.5. Manejo de la memoria no volátil . . . . .                        | 106       |
| 5.3.6. Implementación del sensor amg8833 . . . . .                      | 107       |
| 5.3.7. Implementación del sensor MMA8451 . . . . .                      | 109       |
| 5.3.8. Implementación del buzzer . . . . .                              | 112       |
| 5.3.9. Interrupciones y Deep Sleep . . . . .                            | 113       |
| 5.3.10. Recogida de datos en la ESP8266 . . . . .                       | 117       |
| 5.3.11. Recogidas de datos programadas . . . . .                        | 119       |



|                                                         |            |
|---------------------------------------------------------|------------|
| 5.4. Plataforma IoT . . . . .                           | 120        |
| 5.4.1. Definición . . . . .                             | 120        |
| 5.4.2. Plataformas IoT más relevantes . . . . .         | 121        |
| 5.4.3. Elección de la Plataforma IoT . . . . .          | 124        |
| 5.5. Visualización de los datos . . . . .               | 125        |
| 5.6. Dispositivo instalado . . . . .                    | 129        |
| <b>6. Validación y evaluación</b>                       | <b>130</b> |
| 6.1. Pruebas del sistema . . . . .                      | 130        |
| 6.1.1. Prueba de funcionamiento . . . . .               | 130        |
| 6.1.2. Prueba con el sensor MQ2 . . . . .               | 130        |
| 6.1.3. Prueba con el sensor MQ135 . . . . .             | 131        |
| 6.2. Pruebas del sensor en reposo y en activo . . . . . | 131        |
| 6.3. Resultados . . . . .                               | 131        |
| 6.3.1. Sistema . . . . .                                | 131        |
| 6.3.2. MQ2 . . . . .                                    | 132        |
| 6.3.3. MQ135 . . . . .                                  | 132        |
| 6.3.4. Sensor en reposo y en activo . . . . .           | 134        |
| <b>7. Conclusiones</b>                                  | <b>136</b> |
| 7.1. Revisión del conocimiento adquirido . . . . .      | 136        |
| 7.2. Trabajo futuro . . . . .                           | 136        |
| <b>8. Bibliografía</b>                                  | <b>139</b> |
| <b>9. Anexo</b>                                         | <b>145</b> |
| 9.1. Manual de instalación . . . . .                    | 145        |
| 9.2. Manual de usuario . . . . .                        | 149        |

## Índice de figuras

|                                                                                                                                                                                                                                                                          |    |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 1. Termostato inteligente [4]. . . . .                                                                                                                                                                                                                                   | 9  |
| 2. Pulseras de actividad [2]. . . . .                                                                                                                                                                                                                                    | 10 |
| 3. Diagrama del modelo en cascada [19] . . . . .                                                                                                                                                                                                                         | 21 |
| 4. Diagrama del modelo incremental [1] . . . . .                                                                                                                                                                                                                         | 22 |
| 5. Procesos de Scrum [22] . . . . .                                                                                                                                                                                                                                      | 25 |
| 6. Diagrama de Gantt . . . . .                                                                                                                                                                                                                                           | 29 |
| 7. Esquema del sistema. . . . .                                                                                                                                                                                                                                          | 43 |
| 8. Etapas del sistema. 1a) Mediciones de imágenes termográficas. 1b)<br>Mediciones de gases. . . . .                                                                                                                                                                     | 44 |
| 9. Diagrama de flujo . . . . .                                                                                                                                                                                                                                           | 45 |
| 10. Diagrama de casos de uso . . . . .                                                                                                                                                                                                                                   | 46 |
| 11. Esquema de funcionamiento de la computación en el borde [61]. . . . .                                                                                                                                                                                                | 55 |
| 12. Esquema de funcionamiento de la computación en la nube [61]. . . . .                                                                                                                                                                                                 | 56 |
| 13. Esquema de funcionamiento del Fog Computing [61]. . . . .                                                                                                                                                                                                            | 56 |
| 14. Estructuras de red [49]. . . . .                                                                                                                                                                                                                                     | 58 |
| 15. Placa de desarrollo Lolin32 lite. . . . .                                                                                                                                                                                                                            | 60 |
| 16. Pines Lolin32 lite. . . . .                                                                                                                                                                                                                                          | 61 |
| 17. Placa de desarrollo NodeMcu V3. . . . .                                                                                                                                                                                                                              | 62 |
| 18. Placa de desarrollo Raspberry Pi 3 Model B+ [48]. . . . .                                                                                                                                                                                                            | 63 |
| 19. Sensor Adafruit AMG8833. . . . .                                                                                                                                                                                                                                     | 64 |
| 20. Acelerómetro MMA8451. . . . .                                                                                                                                                                                                                                        | 65 |
| 21. Sensor de gas MQ2. . . . .                                                                                                                                                                                                                                           | 66 |
| 22. Sensibilidad del sensor MQ2 en función del gas [39]. . . . .                                                                                                                                                                                                         | 67 |
| 23. Divisor resistivo. . . . .                                                                                                                                                                                                                                           | 68 |
| 24. Sensor MQ135. . . . .                                                                                                                                                                                                                                                | 69 |
| 25. Sensibilidad del sensor MQ135 en función del gas [60]. . . . .                                                                                                                                                                                                       | 70 |
| 26. Dependencia del MQ-135 por temperatura y humedad; Ro= Resistencia<br>del sensor a 100 ppm de NH <sub>3</sub> en aire a 33% RH y 20 C; RS=<br>Resistencia del sensor con 100 ppm de NH <sub>3</sub> en el aire a diferentes<br>temperaturas y humedades [60]. . . . . | 70 |

|                                                                                               |     |
|-----------------------------------------------------------------------------------------------|-----|
| 27. Buzzer pasivo. . . . .                                                                    | 71  |
| 28. Primer prototipo. . . . .                                                                 | 72  |
| 29. Esquema de conexiones de la primera iteración. . . . .                                    | 72  |
| 30. Segundo prototipo. . . . .                                                                | 73  |
| 31. Esquema de conexiones de la segunda iteración. . . . .                                    | 73  |
| 32. Tercer prototipo. . . . .                                                                 | 74  |
| 33. Modelo original. . . . .                                                                  | 76  |
| 34. Modelo 3D del módulo principal. . . . .                                                   | 77  |
| 35. Modelo 3D del módulo para el sensor de gas. . . . .                                       | 77  |
| 36. Modelo 3D del módulo principal. . . . .                                                   | 78  |
| 37. Pieza en el software de impresión Ultimaker Cura [3]. . . . .                             | 79  |
| 38. Piezas impresas en 3D. . . . .                                                            | 79  |
| 39. Ventana principal de PlatformIO . . . . .                                                 | 89  |
| 40. Visual Studio Code + PlatformIO IDE . . . . .                                             | 89  |
| 41. Modelo de comunicación MQTT [40] . . . . .                                                | 91  |
| 42. Niveles de los topics . . . . .                                                           | 92  |
| 43. Estructura paquete MQTT [66]. . . . .                                                     | 93  |
| 44. Datos de entrada e imagen obtenida. . . . .                                               | 99  |
| 45. Esquema de conexiones para el sistema maestro-esclavo. . . . .                            | 101 |
| 46. Esquema de conexiones para el sensor amg8833. . . . .                                     | 108 |
| 47. Esquema de conexiones para el sensor mma8451. . . . .                                     | 110 |
| 48. Esquema de conexiones para el buzzer. . . . .                                             | 113 |
| 49. Módulos del chip ESP32 [31]. . . . .                                                      | 114 |
| 50. Distribución de la memoria. . . . .                                                       | 118 |
| 51. Esquema de conexiones para el sensor MQ2. . . . .                                         | 119 |
| 52. Ejemplo de un flow en Node-RED . . . . .                                                  | 122 |
| 53. Dashboard de Thinger.io . . . . .                                                         | 124 |
| 54. Histórico de temperaturas máximas y mínimas registradas por la cámara<br>térmica. . . . . | 125 |
| 55. Registro de uso del inodoro. . . . .                                                      | 126 |
| 56. Último valor raw obtenido por el sensor de gas. . . . .                                   | 126 |
| 57. Histórico de valores recogidos por el sensor de gas. . . . .                              | 126 |

|                                                                                    |     |
|------------------------------------------------------------------------------------|-----|
| 58. Data Bucket del sensor de gas. . . . .                                         | 127 |
| 59. Información del dispositivo. . . . .                                           | 127 |
| 60. Información de la base de datos. . . . .                                       | 128 |
| 61. Dispositivo instalado en el inodoro. . . . .                                   | 129 |
| 62. Sistema de alimentación . . . . .                                              | 129 |
| 63. Panel lateral de Visual Studio Code. . . . .                                   | 145 |
| 64. Barra inferior de VSCode. El botón de <i>upload</i> aparece resaltado. . . . . | 145 |
| 65. Parámetros a configurar. . . . .                                               | 146 |
| 66. Dashboard de Thinger.io. . . . .                                               | 149 |
| 67. Información del dispositivo. . . . .                                           | 150 |
| 68. Información de la base de datos . . . . .                                      | 151 |

## Índice de tablas

|     |                                                                                                           |    |
|-----|-----------------------------------------------------------------------------------------------------------|----|
| 1.  | Requisito funcional 1. . . . .                                                                            | 16 |
| 2.  | Requisito funcional 2. . . . .                                                                            | 16 |
| 3.  | Requisito funcional 3. . . . .                                                                            | 16 |
| 4.  | Requisito funcional 4. . . . .                                                                            | 16 |
| 5.  | Requisito funcional 5. . . . .                                                                            | 17 |
| 6.  | Requisito funcional 6. . . . .                                                                            | 17 |
| 7.  | Requisito funcional 7. . . . .                                                                            | 17 |
| 8.  | Requisito no funcional 1. . . . .                                                                         | 18 |
| 9.  | Requisito no funcional 2. . . . .                                                                         | 18 |
| 10. | Requisito no funcional 3. . . . .                                                                         | 18 |
| 11. | Requisito no funcional 4. . . . .                                                                         | 19 |
| 12. | Requisito no funcional 5. . . . .                                                                         | 19 |
| 13. | Planificación temporal del proyecto. . . . .                                                              | 27 |
| 14. | Tarea nº17 . . . . .                                                                                      | 28 |
| 15. | Fecha de inicio y finalización de las principales tareas del proyecto. . .                                | 30 |
| 16. | Gastos de personal. Se ha tenido en cuenta los gastos tomando de referencias [56] [57] [55] [54]. . . . . | 33 |
| 17. | Costes humanos. . . . .                                                                                   | 36 |
| 18. | Especificaciones del equipo de desarrollo. . . . .                                                        | 37 |
| 19. | Costes Hardware. . . . .                                                                                  | 38 |
| 20. | Costes Software. . . . .                                                                                  | 39 |
| 21. | Costes adicionales. . . . .                                                                               | 40 |
| 22. | Total costes. . . . .                                                                                     | 41 |
| 23. | CDU-01. Visualizar las imágenes térmicas. . . . .                                                         | 47 |
| 24. | CDU-02. Visualizar gráficas. . . . .                                                                      | 48 |
| 25. | CDU-03. Consultar la bases de datos. . . . .                                                              | 49 |
| 26. | CDU-04. Modificar la recogida de muestras. . . . .                                                        | 50 |
| 27. | CDU-05. Modificar las imágenes térmicas. . . . .                                                          | 51 |
| 28. | CDU-06. Levantar la tapa. . . . .                                                                         | 52 |
| 29. | CDU-07. Cerrar la tapa. . . . .                                                                           | 53 |

|                                                                                  |     |
|----------------------------------------------------------------------------------|-----|
| 30. Comparación de bases de datos NoSQL. . . . .                                 | 82  |
| 31. Comparación de bases de datos NoSQL. . . . .                                 | 83  |
| 32. Colección de datos para la entidad THERMAL_DATA. . . . .                     | 84  |
| 33. Colección de datos para la entidad GAS_DATA. . . . .                         | 84  |
| 34. Colección de datos para la entidad USE_DATA. . . . .                         | 85  |
| 35. Colección de datos para la entidad GAS_TEST. PPM: Partes por millón. . . . . | 86  |
| 36. Colección de datos para la entidad GAS_TEST. . . . .                         | 87  |
| 37. Resultado de la prueba de gases con el sensor MQ2. . . . .                   | 132 |
| 38. Resultado de la prueba de gases con el sensor MQ135 . . . . .                | 133 |
| 39. Datos obtenidos en reposo por el sensor MQ2 . . . . .                        | 134 |
| 40. Datos obtenidos en reposo por el sensor MQ2 . . . . .                        | 135 |

# 1. Introducción

En este capítulo se va a dar una visión global del proyecto a desarrollar, así como una idea general de en qué consiste y la motivación que nos ha llevado a escogerlo.

## 1.1. Contextualización del proyecto

En la actualidad, el número de dispositivos conectados está en pleno crecimiento. Esto es debido principalmente al desarrollo de componentes cada vez más pequeños y potentes, y al desarrollo de nuevas redes de comunicaciones. Uno de los principales conceptos que aprovechan este tipo de avances es el Internet de las Cosas (IdC) o Internet of Things (IoT). El IoT es la agrupación e interconexión de objetos cotidianos en una red. Estos objetos tienen asociados una serie de sensores y/o dispositivos que nos permiten monitorizarlos, manipularlos a distancia, realizar un procesamiento inteligente e incluso un accionamiento automático en base a este procesamiento. Por ejemplo, un frigorífico inteligente con múltiples sensores y una placa que lee los valores de estos y los procesa, puede llegar a detectar si el habitante se ha dejado la puerta abierta y avisar mediante una notificación móvil de esta anomalía. Cualquier cosa es susceptible de ser conectada a internet e interactuar sin la intervención humana, es decir, el objetivo es una interacción M2M (machine to machine). A medida que avanza el tiempo, cada vez surgen más áreas en las que es posible aplicar el concepto del IoT. A continuación se muestran las cinco aplicaciones más comunes.

1. **Casa conectada.** Monitorización y gestión de los dispositivos o artefactos y sistemas domésticos de forma remota.



Figura 1: Termostato inteligente [4].

2. **Vestibles y bienestar.** Conjunto de dispositivos como relojes o prendas de vestir que interactúan de forma continua con el usuario.



Figura 2: Pulseras de actividad [2].

3. **Ciudades inteligentes.** Proporcionan servicios e información a los residentes y organizaciones públicas para mejorar la calidad de vida. Un ejemplo de este tipo de aplicación es un sistema de alumbrado público, que se regula automáticamente para ahorrar energía.
4. **Internet industrial.** Se utilizan sensores para monitorizar procesos industriales desarrollados por dispositivos robóticos. Este tipo de aplicación va enfocada a la supervisión de los procesos industriales automáticamente.
5. **Salud conectada.** Utilización de sensores y actuadores en su entorno y en los propios pacientes para controlar su seguimiento y mejorar su calidad de vida. Una de las aplicaciones más prometedoras en este ámbito son los medidores de glucosa conectados, los cuales permiten una monitorización continua de los niveles de azúcar en sangre de los pacientes.

Este trabajo fin de grado se va a centrar en el ámbito de la salud conectada. Aunque se explicará con más detalle en secciones posteriores, el objetivo es monitorizar las deposiciones de un habitante como indicador de salud.



## **1.2. Propósito general y objetivos**

### **1.2.1. Objetivo general**

Nuestro objetivo es monitorizar gases y temperatura de las deposiciones que puedan indicar algún tipo de anomalía. Para ello se diseñará y construirá un prototipo de sistema IoT que se encargará de monitorizar las deposiciones de los habitantes. Todo esto vendrá acompañado de una herramienta de visualización en tiempo real que permita consultar de una forma sencilla los datos recogidos por el sistema.

### **1.2.2. Objetivos específicos**

Teniendo en cuenta la propuesta anterior podemos definir los objetivos específicos:

- Revisar y estudiar los conceptos asociados a las tecnologías implicadas del proyecto.
- Analizar las distintas placas de desarrollo existentes para el desarrollo de dispositivos IoT, y seleccionar la más adecuada para nuestro proyecto.
- Construir el dispositivo IoT mediante la placa de desarrollo seleccionada y los diferentes elementos electrónicos necesarios, tales como sensores y actuadores, para llevar a cabo el proyecto.
- Desarrollar un sistema sumidero que recolecte los datos recogidos por los sensores para su posterior tratamiento.
- Diseñar un software que permita transformar los valores de temperatura recogidos por los sensores en imágenes térmicas.
- Diseñar el modelo de datos y estudiar los diferentes tipos de bases de datos existentes para seleccionar la más adecuada para nuestro proyecto.
- Estudiar las distintas plataformas IoT existentes, y seleccionar una de ellas para utilizarla como herramienta de visualización de los datos recopilados por los sensores.
- Proporcionar un método que permita etiquetar los datos para realizar futuros dataset.

- Realizar un estudio sobre los valores que proporcionan distintos sensores con el fin de determinar cual se comporta mejor.
- Evaluar el comportamiento del sensor de gas entre situaciones de reposo y activas.
- Diseñar e imprimir una carcasa 3D que permita fijar el dispositivo de manera más eficaz para poder comercializarlo y usarlo en entornos reales en prototipos futuros.

### **1.3. Revisión de trabajos existentes**

Antes de continuar con los resultados esperados de este proyecto, vamos a realizar una revisión de los trabajos existentes. Dado que el objetivo es la monitorización de las deposiciones, la revisión se ha enfocado a posibles métodos de detección visuales y al análisis de los gases producidos por el organismo.

Un artículo [37], mostró la posibilidad de detectar compuestos orgánicos volátiles (COV) relacionados con el cáncer de colon. Estos COVs pueden ser detectados a través del aliento o los gases del intestino. Para ello se utilizaron una serie de sensores MOX (Metal OXide), que eran capaces de detectar dichos COVs en condiciones de baja y alta humedad. Además, estos sensores podían discernir los COVs objetivos de las interferencias incluso en concentraciones bajas.

Para determinar el tipo de sensores comerciales que podrían ser viables para nuestro objetivo, se realizó una revisión de varios trabajos, entre los que se encuentra un TFG [38]. En este trabajo se expone de manera detallada las principales características de los sensores, así como una guía detallada para extraer las ecuaciones y valores asociados necesarios para calibrar los distintos tipos de sensores. A partir de este trabajo, se ha determinado que es posible utilizar la serie de sensores MQ en el proyecto a tratar.

Respecto a los métodos de detección visuales, la búsqueda de información se ha centrado principalmente en las cámaras térmicas. En numerosos artículos [34] [32] se

muestra la posibilidad de utilizar este tipo de cámaras para la detección de multitud de problemas relacionados con la salud. Además, este tipo de cámaras permiten una mayor anonimidad en comparación con las cámaras tradicionales.

## 1.4. Resultados esperados

Al finalizar el desarrollo e implementación de la aplicación y la construcción del prototipo, se tratarán de lograr los siguientes objetivos:

1. Desarrollar un prototipo funcional, que permita de forma sencilla, rápida y eficaz, recoger los valores de los sensores, almacenarlos y visualizar la información en tiempo real.
2. Realizar un estudio que determine si es posible detectar variaciones significativas mediante el sensor de gas cuando la persona a monitorizar realiza las deposiciones.

## 1.5. Estructura de la memoria

En esta sección, se va a explicar la estructura de la memoria. Este documento consta en total de nueve capítulos:

- **Capítulo 1. Introducción:** Presenta el proyecto y su contexto.
- **Capítulo 2. Análisis:** Especificación sobre las necesidades y requisitos a cubrir por el sistema.
- **Capítulo 3. Planificación:** Organiza la distribución temporal del proyecto en función de los plazos y las tareas a realizar, y analiza los costes asociados a la realización del proyecto.
- **Capítulo 4. Diseño:** Contiene los diseños de las distintas partes que forman el proyecto.
- **Capítulo 5. Implementación:** Se explica de qué modo se ha implementado la totalidad del sistema.

- **Capítulo 6. Validación y evaluación:** Pruebas realizadas para la validación del proyecto y resultados obtenidos.
- **Capítulo 7. Conclusiones:** Revisión del conocimiento adquirido y trabajo futuro.
- **Capítulo 8. Bibliografía:** Incluye las referencias usadas en la elaboración de la memoria y en el desarrollo y preparación del proyecto.
- **Capítulo 9. Anexo:** Contiene el manual de usuario y el manual de instalación.

## **2. Análisis**

En este apartado se identificarán y desarrollarán los distintos aspectos y elementos esenciales que se tendrán en cuenta en las fases de planificación, diseño e implementación.

### **2.1. Problemas a abordar**

El prototipo debe ser capaz de recoger los datos de tal forma que requiera la mínima interacción por parte de la persona a monitorizar y enviar dichos datos a un servidor para su procesamiento y posterior almacenaje de la misma. Además, el sistema deberá de contar con un apartado de visualización de los datos.

### **2.2. Análisis de requisitos**

Durante el desarrollo de este apartado, recogeremos y analizaremos diferentes requisitos funcionales y no funcionales que ha de cumplir nuestro prototipo. Tendremos en cuenta que un requisito funcional define tanto características a nivel de software, como a nivel de componentes. De igual manera, los requisitos no funcionales describen características de funcionamiento. Es importante tener en cuenta que estos requisitos se corresponden con los iniciales, y que al haber escogido un enfoque iterativo, estos requisitos son susceptibles de sufrir cambios durante las siguientes fases del desarrollo software.

#### **2.2.1. Requisitos funcionales**

Como hemos indicado anteriormente, los requisitos funcionales son aquellos que se relacionan directamente con las funcionalidades u objetivos del sistema. En este caso, los requisitos establecidos son los definidos en las Tablas 1, 2, 3, 4, 5, 6, 7.

| Código aplicado | RF-1                                                              |
|-----------------|-------------------------------------------------------------------|
| Nombre          | Monitorización de temperatura                                     |
| Descripción     | El dispositivo será capaz de registrar los valores de temperatura |

Tabla 1: Requisito funcional 1.

| Código aplicado | RF-2                                                            |
|-----------------|-----------------------------------------------------------------|
| Nombre          | Monitorización de gases                                         |
| Descripción     | El dispositivo será capaz de registrar los valores de los gases |

Tabla 2: Requisito funcional 2.

| Código aplicado | RF-3                                                                            |
|-----------------|---------------------------------------------------------------------------------|
| Nombre          | Almacenamiento de datos                                                         |
| Descripción     | El sistema debe ser capaz de almacenar los datos generados en una base de datos |

Tabla 3: Requisito funcional 3.

| Código aplicado | RF-4                                                                   |
|-----------------|------------------------------------------------------------------------|
| Nombre          | Recolección de datos autónoma                                          |
| Descripción     | El prototipo deberá realizar la recogida de datos de manera automática |

Tabla 4: Requisito funcional 4.

| Código aplicado | RF-5                                                                                     |
|-----------------|------------------------------------------------------------------------------------------|
| Nombre          | Visualización de los datos                                                               |
| Descripción     | El sistema debe de permitir visualizar gráficamente los datos recogidos por el prototipo |

Tabla 5: Requisito funcional 5.

| Código aplicado | RF-6                                                                                                                     |
|-----------------|--------------------------------------------------------------------------------------------------------------------------|
| Nombre          | Generación imágenes térmicas                                                                                             |
| Descripción     | El sistema debe ser capaz de generar imágenes térmicas a partir de los datos obtenidos, para facilitar la interpretación |

Tabla 6: Requisito funcional 6.

| Código aplicado | RF-7                                                                |
|-----------------|---------------------------------------------------------------------|
| Nombre          | Monitorización programada                                           |
| Descripción     | El dispositivo debe tomar muestras de datos en intervalos definidos |

Tabla 7: Requisito funcional 7.

### 2.2.2. Requisitos no funcionales

Los requisitos no funcionales están principalmente orientados a proporcionar capacidades o características adicionales tales como rendimiento, estabilidad, eficiencia e integridad de datos entre otros muchos. Estos requisitos no funcionales ayudan al cumplimiento de los requisitos funcionales. En este caso, los requisitos establecidos son los definidos en las Tablas 8, 9, 10, 11, 12.

| Código aplicado | RNF-1                                                                                     |
|-----------------|-------------------------------------------------------------------------------------------|
| Nombre          | Persistencia de datos                                                                     |
| Descripción     | Se debe garantizar la persistencia de los datos aunque se produzca un fallo en el sistema |

Tabla 8: Requisito no funcional 1.

| Código aplicado | RNF-2                                                                                                                                                             |
|-----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Nombre          | Tiempo de respuesta                                                                                                                                               |
| Descripción     | El tiempo de respuesta del dispositivo respecto a las acciones de la persona a monitorizar debe ser mínimo, a fin de recabar la mayor cantidad de datos posibles. |

Tabla 9: Requisito no funcional 2.

| Código aplicado | RNF-3                                                                                 |
|-----------------|---------------------------------------------------------------------------------------|
| Nombre          | Volumen del dispositivo                                                               |
| Descripción     | El tamaño del dispositivo debe ser contenido para poder integrarlo en un entorno real |

Tabla 10: Requisito no funcional 3.



| Código aplicado | RNF-4                                                                                     |
|-----------------|-------------------------------------------------------------------------------------------|
| Nombre          | Eficiencia energética                                                                     |
| Descripción     | El prototipo funciona con baterías, por tanto hay que maximizar la duración de las mismas |

Tabla 11: Requisito no funcional 4.

| Código aplicado | RNF-5                                                      |
|-----------------|------------------------------------------------------------|
| Nombre          | Anonimidad de los datos                                    |
| Descripción     | Los datos recogidos no deben contener información sensible |

Tabla 12: Requisito no funcional 5.

### **3. Planificación**

En esta planificación se abordarán las estimaciones de tiempo necesarias para el prototipo y la aplicación, las estimaciones de costes, recursos humanos, recursos materiales y recursos software.

Dichas estimaciones serán realizadas en base al intervalo de tiempo comprendido entre la fecha de inicio del proyecto y la fecha de entrega del mismo.

El proyecto será dividido en etapas o fases de desarrollo, las cuales podrán englobar a su vez diferentes subtarefas. Cada fase y su ejecución determinarán la correcta evolución del proyecto.

Los costes totales que se exponen a continuación, se corresponden al desarrollo del proyecto e implementación del mismo y con perspectiva a ser comercializado como producto.

#### **3.1. Metodología**

El proceso de creación del software es una labor dependiente del juicio humano, y por tanto, presenta una inmensa diversidad de opciones. Es por esta razón que no existe un proceso ideal de desarrollo de software. Este proceso depende en mayor medida del tamaño del proyecto, de su tipo y de la organización que lo desarrolla.

##### **3.1.1. Principales metodologías**

Las principales metodologías existentes para el desarrollo software son el modelo en cascada, el modelo incremental y el desarrollo ágil de software, las cuales se explicarán a continuación.

## El modelo en cascada

El modelo en cascada [19] o también denominado ciclo de vida clásico, es el paradigma de desarrollo de software más antiguo. Se caracteriza por dividir los procesos de desarrollo en sucesivas fases del proyecto. Cada una de estas fases sirven como hipótesis de partida de la siguiente, pero solo se pueden ejecutar una única vez. Aunque existen diversas versiones del modelo, los más habituales cuentan con 5 fases.

- **Análisis.** Planificación, análisis y especificación de los requisitos.
- **Diseño.** Diseño y especificación del sistema.
- **Implementación.** Programación y pruebas unitarias.
- **Verificación.** Integración de sistemas, pruebas de sistema y de integración.
- **Mantenimiento.** Entrega, mantenimiento y mejora.



Figura 3: Diagrama del modelo en cascada [19]

Las principales ventajas de utilizar este modelo residen en que los costes y la carga de trabajo se pueden estimar al inicio. Consta de una estructura sencilla gracias a unas fases del proyecto bien diferenciadas. Los proyectos que se estructuran en base al modelo en cascada se pueden representar cronológicamente de una manera más sencilla.

Los principales inconvenientes de este tipo de modelo radican en que el usuario final no se integra en el proceso de producción hasta que no finaliza la programación. En proyectos complejos es difícil realizar la división en fases tan diferenciadas. Ocasionalmente los fallos solo se detectan una vez ha finalizado el proceso de desarrollo.

### El modelo incremental [35]

Este modelo descompone un proyecto en una sucesión de agregados denominados incrementos, de tal modo que cada entrega es producto operativo pero incompleto.

Los principales requisitos del proyecto se incluyen en los primeros incrementos. Una vez se ha empezado el desarrollo de un incremento, sus requisitos no se alteran hasta que este finalice. Los cambios planteados podrán ser introducidos en la planificación de un incremento posterior.

Cada incremento deberá ser desarrollado en un tiempo previamente establecido y breve.

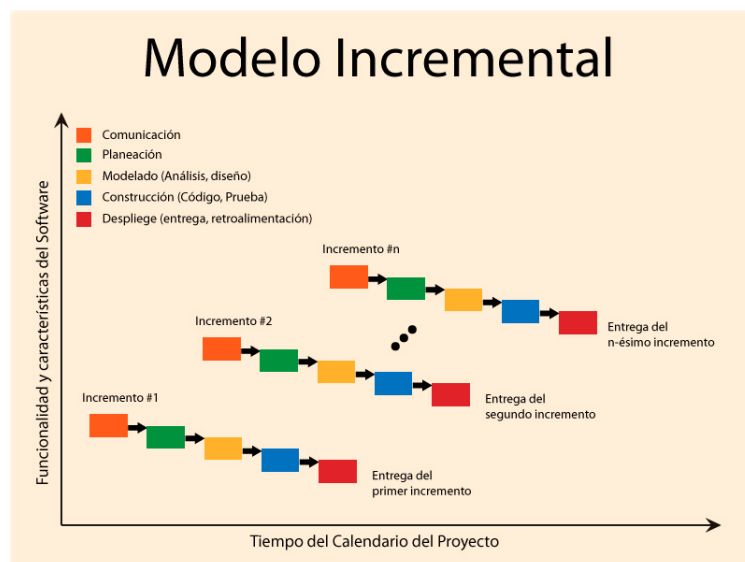


Figura 4: Diagrama del modelo incremental [1]

Todas estas características posicionan al modelo incremental por encima del desarrollo en cascada, puesto que reduce el riesgo de fallo del proyecto al permitir

al cliente probar los diferentes prototipos y realizar cambios en los requisitos.

## Desarrollo ágil de software

Las metodologías ágiles de desarrollo software [11] buscan proporcionar en poco tiempo pequeñas partes del sistema software en funcionamiento al cliente.

### ■ Extreme Programming.

El eXtreme Programming (XP) [36] es considerada la aplicación más radical del desarrollo ágil de software. Al igual que otros métodos ágiles, XP divide el proyecto en procesos iterativos.

#### Roles

- **Cliente.** El cliente es considerado como parte del equipo, y es imprescindible que siempre haya algún representante presente. Se encargará de marcar la prioridad de los ámbitos parciales.
- **Desarrollador.** En este rol no solo se incluyen programadores, sino también otras personas implicadas en la creación, según las necesidades del proyecto.
- **Director.** Es el encargado de la comunicación entre los desarrolladores y los clientes, organizando las reuniones entre ambas partes.
- **Coach.** Su función es instruir a todo el equipo (incluido el cliente) en el uso del extreme programming. Puede estar presente durante todas las fases del desarrollo para ayudar a resolver dudas.

### ■ Scrum

Scrum [52] es una metodología de trabajo incremental e iterativa para la gestión de proyectos, usada principalmente para el desarrollo software.

Dentro del marco de trabajo Scrum existen tres roles básicos:

- **Propietario del producto (Product owner).** Es la persona que determina las prioridades del proyecto y representa a los usuarios o a la empresa.
- **Equipo de desarrollo.** Es el equipo que se encargará de desarrollar el producto.
- **Facilitador de proyectos (Scrum master).** Es la persona encargada de gestionar la correcta aplicación de la metodología en el proyecto. También se encarga de facilitar reuniones y eventos en caso de ser necesario.

Scrum incorpora una serie de eventos y artefactos para poder gestionar la metodología.

### Eventos Scrum

- **Sprint.** Los Sprints son eventos de duración fija, con una duración de un mes o menos.
- **Sprint Planning.** Es la reunión previa para planificar el sprint. En esta reunión se determina todo el trabajo que se realizará durante el Sprint.
- **Daily Scrum.** Es una reunión diaria de 15 minutos de duración, en la que el equipo de desarrollo se reúne para revisar el trabajo realizado el día anterior y el que se realizará ese día.
- **Sprint Review.** En esta reunión se presenta todo el trabajo realizado durante el Sprint. Basándose en esta información se decidirá que hacer a continuación. El Product Backlog puede ser modificado para añadir nuevas tareas.
- **Sprint Retrospective.** En esta reunión el equipo determina lo que ha ido bien durante el Sprint, los problemas surgidos, y si se han solucionado o no dichos problemas.

## Artefactos Scrum

- **Product Backlog.** Es la principal fuente de información sobre el producto en Scrum, es una lista que contiene cualquier tipo de trabajo que haya que hacer en el producto. El Product Backlog es gestionado exclusivamente por el Product Owner.
- **Sprint Backlog.** Es una lista de los elementos que se deben trabajar durante el Sprint. Está gestionado por el equipo de desarrollo.
- **Incremento.** Es el resultado del Sprint, es el conjunto de todas las tareas que se hayan desarrollado durante el Sprint.

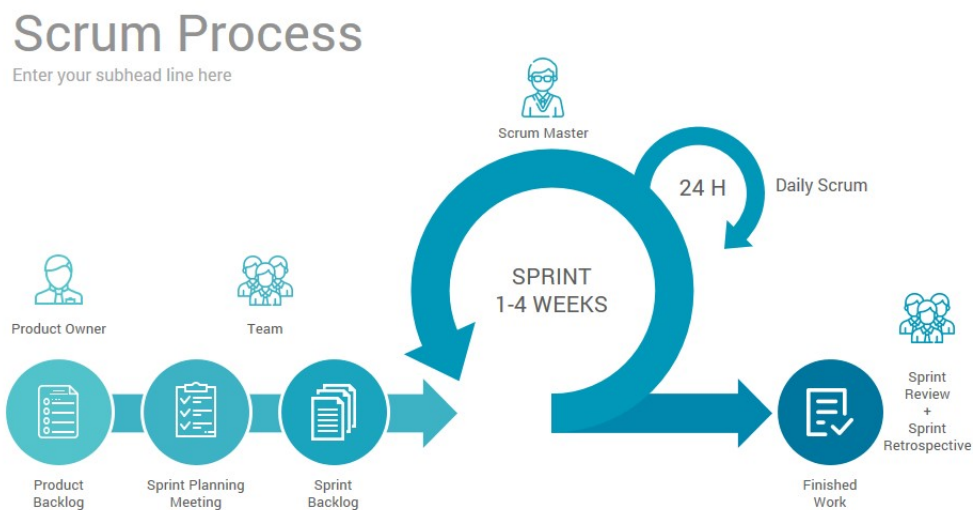


Figura 5: Procesos de Scrum [22]

### 3.1.2. Elección de la metodología

Como hemos podido ver anteriormente, existen multitud de metodologías que podemos utilizar a la hora de desarrollar software. En nuestro caso nos hemos decantado por utilizar una metodología de desarrollo ágil, en particular Scrum. Esto es debido a que Scrum nos permite tener un mayor control sobre el desarrollo de nuestro proyecto, aumentando las probabilidades de éxito.

Scrum como hemos mencionando anteriormente es una metodología de desarrollo enfocada a equipos. Esto supone una serie de limitaciones a la hora de aplicarlo a un

solo individuo como es nuestro caso.

- Dado que el equipo de desarrollo está conformado unicamente por un desarrollador, no es posible realizar varias tareas simultáneamente.
- Reducimos la duración del Sprint a dos semanas. De este modo tenemos un mayor control sobre el trabajo a realizar.
- El tutor, José Luis López Ruiz, ha supervisado el desarrollo de los diferentes Sprints.

### **3.2. Planificación de tareas**

Las tareas que se presentan a continuación, son las consideradas como fases críticas del proyecto. Para ilustrarlas, se ha realizado una Tabla 13 donde se especifica la tarea, así como su duración estimada.



| Tarea                                        | Duración (Horas) |
|----------------------------------------------|------------------|
| Búsqueda de información sobre IoT            | 15               |
| Estudio de tecnologías de comunicación       | 15               |
| Revisión de trabajos existentes              | 20               |
| Análisis de requisitos                       | 15               |
| Diseño de la base de datos                   | 5                |
| Desarrollo del dispositivo IoT               | 100              |
| Desarrollo de la aplicación de visualización | 10               |
| Diseño de los modelos 3D                     | 15               |
| Implementación de la base de datos           | 20               |
| Pruebas                                      | 20               |
| Corrección de errores                        | 15               |
| Redacción de la memoria                      | 50               |
| Total                                        | 300              |

Tabla 13: Planificación temporal del proyecto.

Adicionalmente a estas tareas, durante el desarrollo del proyecto se utilizaron diferentes subtareas. Estas subtareas poseían una serie de características, como pueden ser la prioridad de la tarea, su duración estimada y la parte del sistema a la que afecta. A continuación se muestra un ejemplo.

| Tarea nº17       | Implementación de la cámara térmica |
|------------------|-------------------------------------|
| Prioridad        | Alta                                |
| Horas estimadas  | 8                                   |
| Sistema afectado | Dispositivo                         |
| Finalidad        | Recoger imágenes térmicas           |

Tabla 14: Tarea nº17

Las prioridades de las tareas se encargan de garantizar que se completan según su importancia. Si una tarea no es completada en el tiempo previsto, su nivel de prioridad asciende. Se utilizaron tres niveles de prioridad, baja, media y alta.

### 3.3. Diagrama de Gantt

El diagrama de Gantt es una herramienta para la gestión de proyectos en la que se recoge la planificación de un proyecto [24]. Normalmente está formado por dos secciones, en la parte izquierda se incluye una lista con las diferentes tareas de nuestro proyecto y, en la parte derecha, un cronograma de barras que representa el trabajo. Los diagramas de Gantt pueden incluir información relacionada con las fechas de inicio y finalización de las tareas, los diferentes hitos, las dependencias existentes entre las tareas y las personas asignadas a dichas tareas.

Los diagramas de Gantt nos facilitan la creación de planes complejos, especialmente en aquellos en los que participan varios equipos y cuyos plazos están sujetos a modificaciones.

A continuación, se va a presentar el diagrama de Gantt asociado a nuestra planificación, que mediante una representación gráfica nos permitirá visualizar de forma más explícita la evolución que debe seguir nuestro proyecto.

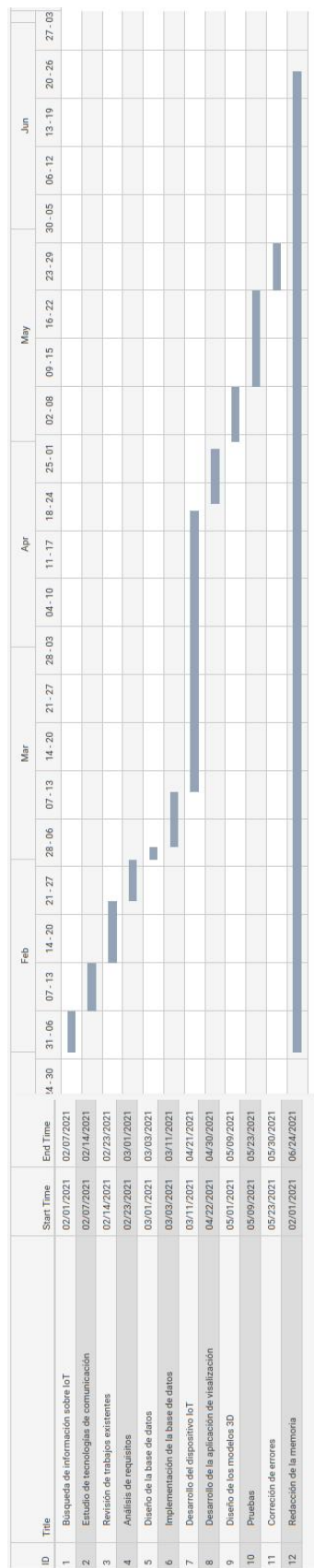


Figura 6: Diagrama de Gantt

| Tarea                                        | Inicio-Finalización   |
|----------------------------------------------|-----------------------|
| Búsqueda de información sobre IoT            | 02/01/2021-02/07/2021 |
| Estudio de tecnologías de comunicación       | 02/07/2021-02/14/2021 |
| Revisión de trabajos existentes              | 02/14/2021-02/23/2021 |
| Análisis de requisitos                       | 02/23/2021-03/01/2021 |
| Diseño de la base de datos                   | 03/01/2021-03/03/2021 |
| Implementación de la base de datos           | 03/03/2021-03/11/2021 |
| Desarrollo del dispositivo IoT               | 03/11/2021-04/21/2021 |
| Desarrollo de la aplicación de visualización | 04/22/2021-04/30/2021 |
| Diseño de los modelos 3D                     | 05/01/2021-05/09/2021 |
| Pruebas                                      | 05/09/2021-05/23/2021 |
| Corrección de errores                        | 05/23/2021-05/30/2021 |
| Redacción de la memoria                      | 02/01/2021-06/24/2021 |

Tabla 15: Fecha de inicio y finalización de las principales tareas del proyecto.

### 3.4. Estimación de costes

En este apartado realizaremos un análisis de los costes de desarrollo y ejecución del proyecto. Se tendrán en cuenta los costes asociados al proyecto de manera directa e indirecta.

Las estimaciones de costes las realizaremos en base al tiempo de duración y al hardware necesario estimado del proyecto.

### **3.4.1. Costes humanos**

Los costes o recursos humanos se corresponden al número y tipo de trabajadores que necesitaremos para la realización de nuestro proyecto. Nuestro proyecto se divide en diferentes tareas, las cuales serán realizadas por distintos perfiles de trabajadores.

A la hora de calcular los costes asociados a cada uno de estos trabajadores tendremos que tener en cuenta el coste de emplear al trabajador, así como el tiempo que tenga que dedicar. En base a las tareas de nuestro proyecto, se calculará el número de empleados necesarios, y una vez realizado esto, se determinará el sueldo a pagar a cada uno.

Hay que tener en cuenta que no todo el año es laborable, por lo que los siguientes conjuntos de días no se podrá trabajar.

- Un mínimo de 30 días de vacaciones por año.
- 48 sábados al año (excluyendo los incluidos en las vacaciones).
- 48 domingos al año (excluyendo los incluidos en las vacaciones).
- 14 días festivos por año.

Esto resulta en un total de 140 días no laborables, los cuales restados al número de días que contiene un año (365), nos deja un total de 225 días hábiles al año por trabajador. Además, se supondrá que los trabajadores trabajan a jornada completa, es decir, 8 horas al día. Esto nos deja un total de 1800 horas hábiles.

Los perfiles necesarios para realizar el proyecto son los siguientes:

- Jefe de proyectos.
- Analista programador.
- Programador senior.
- Programador junior.

A continuación veremos cual será el coste para cada uno de ellos, aplicando Seguridad Social.

| Concepto                              | Jefe de proyectos | Analista programador | Programador senior | Programador junior |
|---------------------------------------|-------------------|----------------------|--------------------|--------------------|
| Salario mensual bruto                 | 2650€             | 2000€                | 2200€              | 1800€              |
| Pagas extras                          | 2                 | 2                    | 2                  | 2                  |
| Base de cotización                    | 3091,66€          | 2333,33€             | 2566,66€           | 2100€              |
| Contingencias comunes:<br>23,50 %     | 729,63€           | 550,66€              | 605,73€            | 495,6€             |
| Desempleo:<br>5,5 %                   | 214,59€           | 128,33€              | 141,16€            | 115,5€             |
| Contingencias profesionales:<br>1,5 % | 58,52€            | 34,99€               | 38,49€             | 31,5€              |
| Formación profesional:<br>0,60 %      | 23,40€            | 13,99€               | 15,39€             | 12,6€              |
| Fogasa:<br>0,20 %                     | 7,8€              | 4,66€                | 5,13€              | 4,2€               |
| Total Seguridad Social                | 1033,94€          | 732,63€              | 805,9€             | 659,4€             |
| Total mensual                         | 3683,94€          | 2732,63€             | 3005,9€            | 2459,4€            |

|                        |                  |                  |                 |                 |
|------------------------|------------------|------------------|-----------------|-----------------|
| Sueldo anual           | 37100€           | 28000€           | 30800€          | 25200€          |
| Seguridad Social anual | 12407,28€        | 8791,56€         | 9670,8€         | 7912,8€         |
| <b>Total anual</b>     | <b>49507,28€</b> | <b>36791,56€</b> | <b>40470,8€</b> | <b>33112,8€</b> |

Tabla 16: Gastos de personal. Se ha tenido en cuenta los gastos tomando de referencias [56] [57] [55] [54].

### Jefe de proyectos

Salario: 2650€ mensuales brutos, con 2 pagas extraordinarias [56].

Base de cotización:  $(2650 \times 14) / 12 \rightarrow 3091,66€$

Gastos Seguridad Social:

- Contingencias comunes: 23,60 %  $\rightarrow 729,63€$
- Prestación por desempleo ( contrato indefinido ): 5,5 %  $\rightarrow 214,59€$
- Contingencias profesionales (programación, consultoría, y otras actividades relacionadas con la informática): 1,5 %  $\rightarrow 58,52€$
- Formación profesional: 0,60 %  $\rightarrow 23,40€$
- FOGASA: 0,20 %  $\rightarrow 7,8€$

Total Seguridad Social  $\rightarrow 1033,94€$

Total mensual  $\rightarrow 3683,94€$

Sueldo  $\rightarrow 2650 \times 14$  pagas  $\rightarrow 37100€$

Seguridad Social  $\rightarrow 1033,94 \times 12 \rightarrow 12407,28€$

Total anual -> **49507,28€**

### **Analista programador**

Salario: 2000€ mensuales brutos, con 2 pagas extraordinarias [57].

Base de cotización:  $(2000 \times 14) / 12 \rightarrow 2333,33€$

Gastos Seguridad Social:

- Contingencias comunes: 23,60 % -> 550,66€
- Prestación por desempleo ( contrato indefinido ): 5,5 % -> 128,33€
- Contingencias profesionales (programación, consultoría, y otras actividades relacionadas con la informática): 1,5 % -> 34,99€
- Formación profesional: 0,60 % -> 13,99€
- FOGASA: 0,20 % -> 4,66€

Total Seguridad Social -> 732,63€

Total mensual -> 2732,63€

Sueldo -> 2000 x 14 pagas -> 28000€

Seguridad Social -> 732,63 x 12 -> 8791,56€

Total anual -> **36791,56€**

### **Programador senior**

Salario: 2200€ mensuales brutos, con 2 pagas extraordinarias [55].

Base de cotización:  $(2200 \times 14) / 12 \rightarrow 2566,66€$



Gastos Seguridad Social:

- Contingencias comunes: 23,60 % -> 605,73€
- Prestación por desempleo ( contrato indefinido ): 5,5 % -> 141,16€
- Contingencias profesionales (programación, consultoría, y otras actividades relacionadas con la informática): 1,5 % -> 38,49€
- Formación profesional: 0,60 % -> 15,39€
- FOGASA: 0,20 % -> 5,13€

Total Seguridad Social -> 805,9€

Total mensual -> 3005,9€

Sueldo -> 2200 x 14 pagas -> 30800€

Seguridad Social -> 805,9 x 12 -> 9670,8€

Total anual -> **40470,8€**

**Programador junior**

Salario: 1800€ mensuales brutos, con 2 pagas extraordinarias [54]

Base de cotización:  $(1800 \times 14) / 12$  -> 2100€

Gastos Seguridad Social:

- Contingencias comunes: 23,60 % -> 495,6€
- Prestación por desempleo ( contrato indefinido ): 5,5 % -> 115,5€
- Contingencias profesionales (programación, consultoría, y otras actividades relacionadas con la informática): 1,5 % -> 31,5€

- Formación profesional: 0,60 % -> 12,6€
- FOGASA: 0,20 % -> 4,2€

Total Seguridad Social -> 659,4€

Total mensual -> 2459,4€

Sueldo -> 1800 x 14 pagas -> 25200€

Seguridad Social -> 659,4 x 12 -> 7912,8€

Total anual -> **33112,8€**

| Puesto profesional   | Costes               | Tiempo de trabajo | Total            |
|----------------------|----------------------|-------------------|------------------|
| Jefe de proyectos    | 49507,28€<br>anuales | 2 meses           | 7367,88€         |
| Analista programador | 36791,56€<br>anuales | 2 meses           | 5465,26 €        |
| Programador senior   | 40470,8€ anuales     | 2 meses           | 6011,8 €         |
| Programador junior   | 33112,8€ anuales     | 2 meses           | 4918,8€          |
| <b>Total</b>         |                      |                   | <b>23763,74€</b> |

Tabla 17: Costes humanos.

### 3.4.2. Costes de hardware y software

En esta sección se reflejarán los costes de Hardware (materiales, equipos, instalación, sensores, etc) que requerirá el desarrollo del proyecto. Así como los costes software (Sistema operativo o programas ) necesarios para la realización del proyecto.

El proyecto se ha realizado en un equipo que tiene las siguientes especificaciones:

| Componentes             | Especificaciones           |
|-------------------------|----------------------------|
| Procesador              | Intel core i7-5500U 2.4GHz |
| Memoria RAM             | 12 GB                      |
| Gráfica integrada       | Gráficos HD Intel 5500     |
| Gráfica dedicada        | Nvidia GeForce GTX 850M    |
| Unidad de estado sólido | Kingston A400 480GB        |
| Disco duro              | 1 TB                       |

Tabla 18: Especificaciones del equipo de desarrollo.

| Hardware               | Unidades | Precio unitario |
|------------------------|----------|-----------------|
| Ordenador personal     | 1        | 820€            |
| ESP8266                | 1        | 7,99€           |
| ESP32                  | 1        | 8,99€           |
| Protoboard             | 2        | 4,49€           |
| Raspberry Pi Model 3B+ | 1        | 44,95€          |
| Cables                 | 30       | 0,1€            |
| Batería                | 1        | 12,99€          |
| MQ2                    | 1        | 4,99€           |
| MQ135                  | 1        | 4,99€           |
| AMG8833                | 1        | 49,99€          |
| Resistencias           | 3        | 0,1€            |
| <b>Total</b>           |          | <b>967,17€</b>  |

Tabla 19: Costes Hardware.

| Software                                     | Unidades | Precio unitario |
|----------------------------------------------|----------|-----------------|
| Sistema operativo elementary OS 5.1.7 «Hera» | 1        | 0€              |
| Visual Studio Code + PlatformIO              | 1        | 0€              |
| Python 3.6.9 y librerías de Python           | 1        | 0€              |
| TeXstudio                                    | 1        | 0€              |
| MongoDB                                      | 1        | 0€              |
| MongoDB Compass                              | 1        | 0€              |
| Eclipse Mosquitto                            | 1        | 0€              |
| Thinger.io                                   | 1        | 0€              |
| FreeCAD                                      | 1        | 0€              |
| GitLab Ujaen                                 | 1        | 0€              |
| Ultimaker Cura                               | 1        | 0€              |
| Visual Paradigm                              | 1        | 0€              |
| <b>Total</b>                                 |          | <b>0€</b>       |

Tabla 20: Costes Software.

### 3.4.3. Costes adicionales

En el apartado de costes adicionales se incluirán los costes de los servicios derivados del desarrollo en el periodo de tiempo especificado. Los servicios que se contemplan son los siguientes:

| Servicio                                                              | Coste al mes | Tiempo de uso | Coste total  |
|-----------------------------------------------------------------------|--------------|---------------|--------------|
| Servicio de Internet                                                  | 80€          | 2 meses       | 160€         |
| Gastos de contingencias (luz, agua..)                                 | 1248€        | 2 meses       | 2496€        |
| Material fungible (filamento impresora 3D, pegamento termofusible...) | 50€          | 2 meses       | 100€         |
| <b>Total</b>                                                          |              |               | <b>2756€</b> |

Tabla 21: Costes adicionales.

#### 3.4.4. Costes totales

Una vez calculados los costes parciales del proyecto se procederá a calcular el coste total, añadiendo un beneficio estimado del 6 %.

| Categoría          | Coste            |
|--------------------|------------------|
| Costes humanos     | 23763,74€        |
| Costes hardware    | 967,17€          |
| Costes software    | 0€               |
| Costes adicionales | 2756€            |
| Total costes       | 27486,91€        |
| Beneficio (6 %)    | 1649,21€         |
| <b>Total</b>       | <b>29136,12€</b> |

Tabla 22: Total costes.

## 4. Diseño

En este apartado se abordará el diseño software, considerando el hardware del sistema, la comunicación entre los dispositivos, el diseño y construcción de las carcasas para nuestro hardware, y el modelo de datos empleado.

### 4.1. Arquitectura del sistema IoT

En esta sección se explicara la arquitectura del sistema IoT propuesto para este proyecto.

El sistema IoT diseñado está compuesta por una serie de elementos, que se encargan de garantizar que se cumplen los objetivos del proyecto..

#### Elementos

- **Esp32, ESP8266 y sensores:** Estas placas de desarrollo y sensores se encargan de recoger todos los datos necesarios. Dichos datos son enviados mediante el protocolo MQTT (Message Queing Telemetry Transport) [40] al servidor ubicado en la Raspberry Pi. Además, también se envían a una plataforma IoT para permitir una visualización más sencilla de los mismos.
- **Raspberry Pi:** Recibe los datos de las placas de desarrollo mediante el protocolo MQTT. Una vez recibidos, los datos se procesan y son insertados en una base de datos.
- **Base de datos:** Almacena todos los datos provenientes de los sensores, para permitir al usuario/cuidador realizar una revisión posterior de los mismos.
- **Plataforma IoT:** Permite al usuario/cuidador visualizar la información de los sensores, así como consultar el estado de los dispositivos.

Se ha diseñado un esquema, Figura 7, que representa la arquitectura del sistema, con los distintos elementos que lo conforman y su interacción.



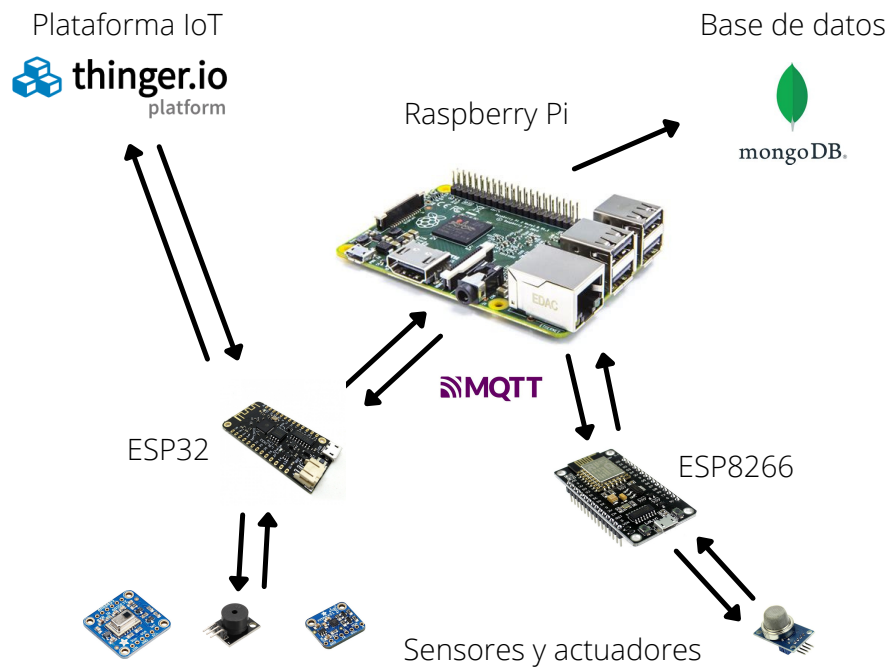


Figura 7: Esquema del sistema.

Además, en la Figura 8, se muestran las distintas fases que se realizan.

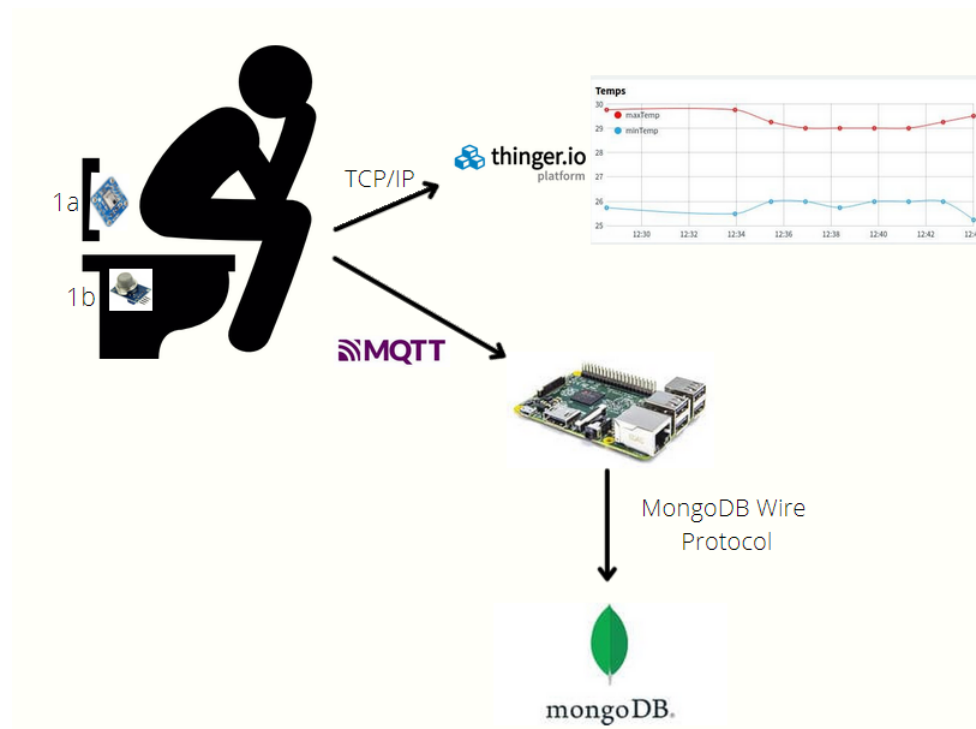


Figura 8: Etapas del sistema. 1a) Mediciones de imágenes termográficas. 1b) Mediciones de gases.

## 4.2. Diseño software

En esta sección abordaremos el diseño de software, que es el proceso por el cual un agente crea una especificación de un artefacto de software, pensado para cumplir unos objetivos determinados.

### 4.2.1. Diagrama de flujo

Un diagrama de flujo [7] es un diagrama que describe un proceso, algoritmo o sistema. En nuestro caso hemos utilizado este tipo de diagrama para describir el comportamiento del dispositivo IoT instalado en la tapa del inodoro, para la monitorización de gases y temperatura de las deposiciones.

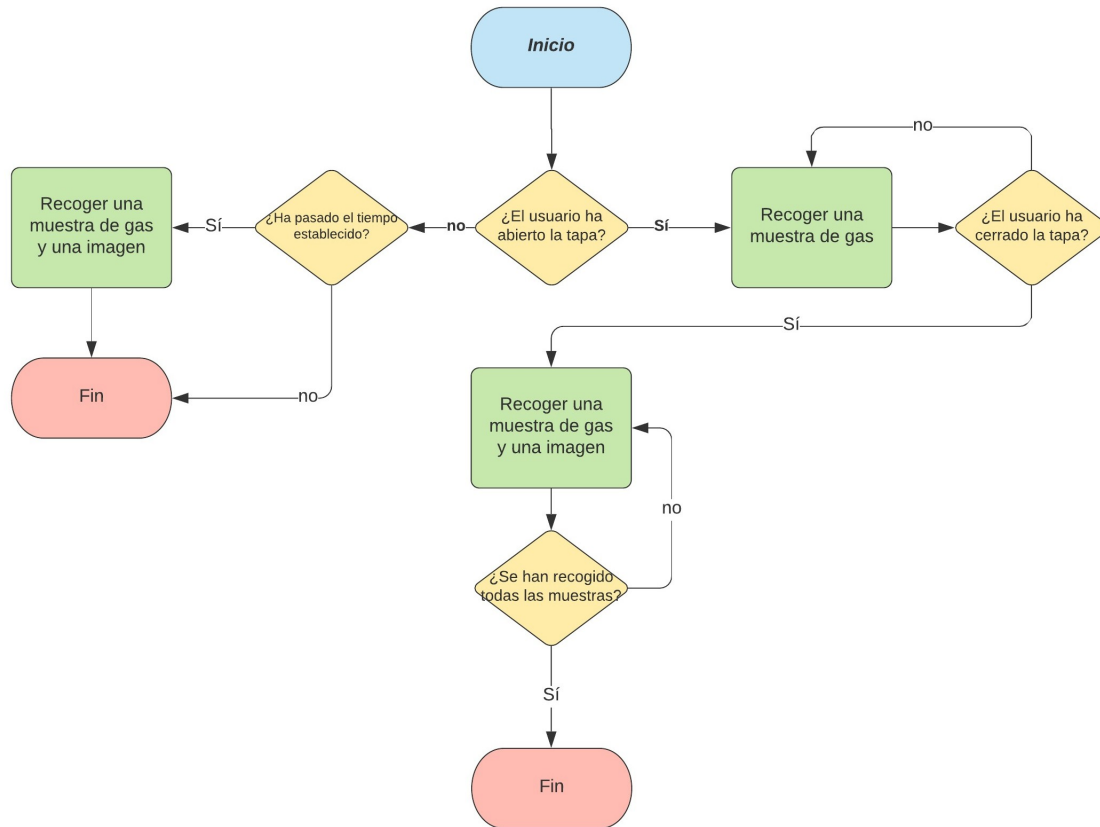


Figura 9: Diagrama de flujo

#### 4.2.2. Casos de uso

En los casos de uso [64] tendremos en cuenta las acciones que cada usuario/cuidador o persona a monitorizar puede realizar dentro del sistema.

#### Diagrama de casos de uso

A continuación se muestran en forma de diagrama, los diferentes casos de uso que hemos identificado durante la planificación del sistema.

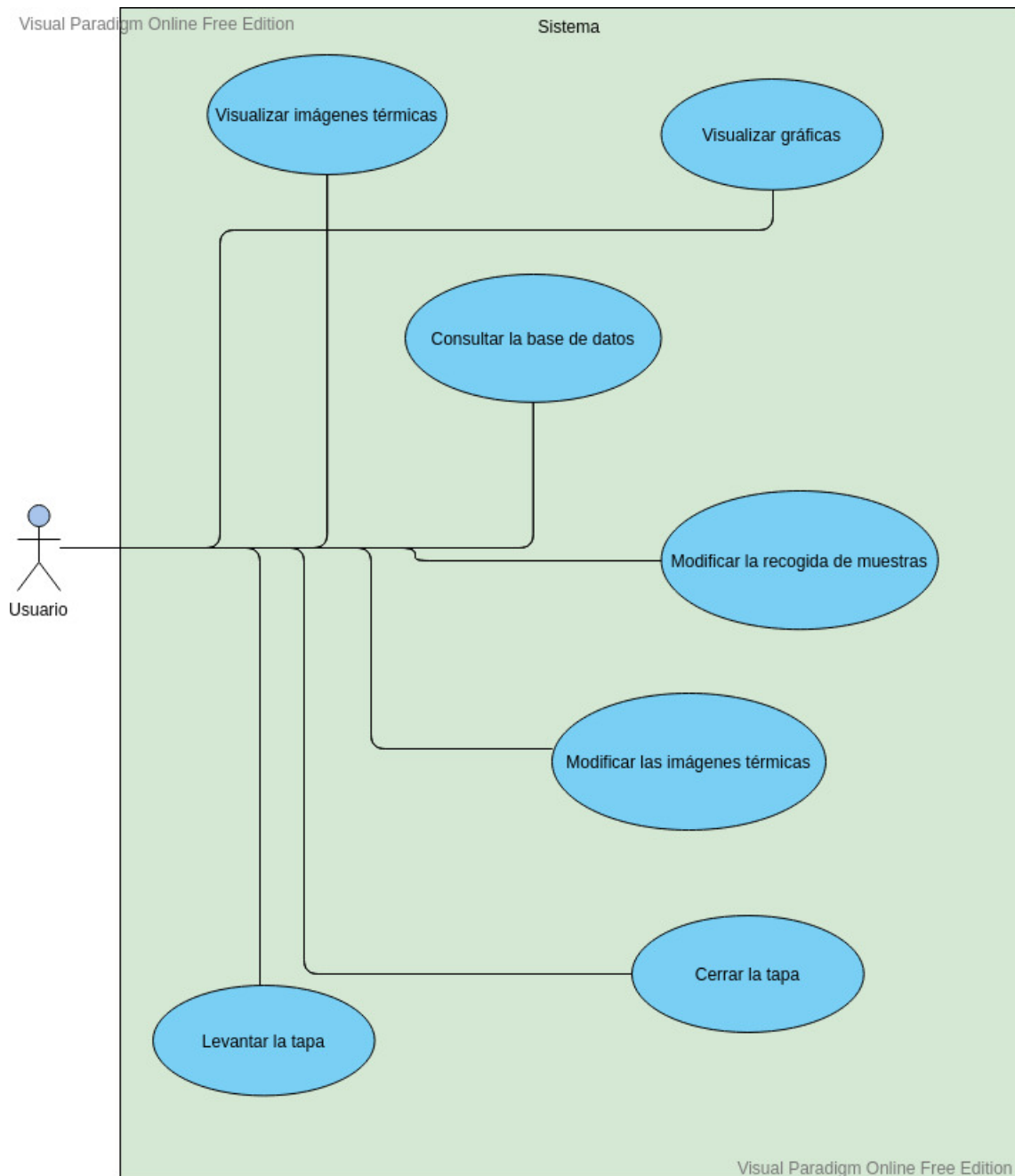


Figura 10: Diagrama de casos de uso

### Definición de casos de uso

A continuación vamos a proceder a detallar los casos de uso de nuestro sistema.

| CDU-01              | Visualizar las imágenes térmicas                                                                                                                                                 |
|---------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Participantes       | Usuario/cuidador                                                                                                                                                                 |
| Descripción         | El usuario/cuidador debe iniciar sesión en la Raspberry Pi. Una vez allí deberá acceder a la carpeta de imágenes para poder visualizarlas.                                       |
| Condición anterior  | Haber generado alguna imagen.                                                                                                                                                    |
| Flujo natural       | <ol style="list-style-type: none"> <li>1. Iniciar sesión en la Raspberry Pi.</li> <li>2. Acceder a la carpeta de imágenes.</li> <li>3. Seleccionar la imagen deseada.</li> </ol> |
| Flujo alternativo   | No existe flujo alternativo.                                                                                                                                                     |
| Condición posterior | Ninguna                                                                                                                                                                          |
| Observaciones       | Si el sistema no ha sido desplegado, la carpeta se encontrará vacía.                                                                                                             |

Tabla 23: CDU-01. Visualizar las imágenes térmicas.

| CDU-02              | Visualizar gráficas                                                                                                                   |
|---------------------|---------------------------------------------------------------------------------------------------------------------------------------|
| Participantes       | Usuario/cuidador                                                                                                                      |
| Descripción         | El usuario/cuidador puede visualizar unas gráficas que contienen datos derivados del uso del dispositivo.                             |
| Condición anterior  | Ninguna                                                                                                                               |
| Flujo natural       | <ol style="list-style-type: none"> <li>1. Iniciar sesión en la plataforma IoT.</li> <li>2. Acceder a la sección Dashboard.</li> </ol> |
| Flujo alternativo   | No existe flujo alternativo.                                                                                                          |
| Condición posterior | Ninguna                                                                                                                               |
| Observaciones       | Las gráficas que se muestran contienen datos de la última semana.                                                                     |

Tabla 24: CDU-02. Visualizar gráficas.

| CDU-03              | Consultar la bases de datos                                                                                                                    |
|---------------------|------------------------------------------------------------------------------------------------------------------------------------------------|
| Participantes       | Usuario/cuidador                                                                                                                               |
| Descripción         | El usuario/cuidador puede consultar la información contenida en la base de datos mediante una aplicación de gestión o mediante consola.        |
| Condición anterior  | Ninguna                                                                                                                                        |
| Flujo natural       | <ol style="list-style-type: none"> <li>1. Iniciar sesión en MongoDB Compass.</li> <li>2. Acceder a la base de datos seleccionada.</li> </ol>   |
| Flujo alternativo   | <ol style="list-style-type: none"> <li>1. Conectarse mediante línea de comandos.</li> <li>2. Realizar consultas a la base de datos.</li> </ol> |
| Condición posterior | Ninguna                                                                                                                                        |
| Observaciones       | Ninguna                                                                                                                                        |

Tabla 25: CDU-03. Consultar la bases de datos.

| CDU-04              | Modificar la recogida de muestras                                                                                                                                                                          |
|---------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Participantes       | Usuario/cuidador                                                                                                                                                                                           |
| Descripción         | El usuario/cuidador puede modificar los parámetros de configuración relacionados con la recogida de datos del dispositivo.                                                                                 |
| Condición anterior  | Ninguna                                                                                                                                                                                                    |
| Flujo natural       | <ol style="list-style-type: none"> <li>1. Iniciar sesión en la Raspberry Pi.</li> <li>2. Acceder a la carpeta que contiene los archivos de configuración.</li> <li>3. Modificar los parámetros.</li> </ol> |
| Flujo alternativo   | No existe flujo alternativo.                                                                                                                                                                               |
| Condición posterior | Ninguna.                                                                                                                                                                                                   |
| Observaciones       | La configuración no se cargará hasta que se vuelva a iniciar la ejecución del programa.                                                                                                                    |

Tabla 26: CDU-04. Modificar la recogida de muestras.



| CDU-05              | Modificar las imágenes térmicas                                                                                                                                                                            |
|---------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Participantes       | Usuario/cuidador                                                                                                                                                                                           |
| Descripción         | El usuario/cuidador puede modificar los parámetros de configuración de la generación de imágenes térmicas.                                                                                                 |
| Condición anterior  | Ninguna                                                                                                                                                                                                    |
| Flujo natural       | <ol style="list-style-type: none"> <li>1. Iniciar sesión en la Raspberry Pi.</li> <li>2. Acceder a la carpeta que contiene los archivos de configuración.</li> <li>3. Modificar los parámetros.</li> </ol> |
| Flujo alternativo   | No existe flujo alternativo.                                                                                                                                                                               |
| Condición posterior | Ninguna                                                                                                                                                                                                    |
| Observaciones       | La configuración no se cargará hasta que se vuelva a iniciar la ejecución del programa.                                                                                                                    |

Tabla 27: CDU-05. Modificar las imágenes térmicas.

| CDU-06              | Levantar la tapa                                                                                                          |
|---------------------|---------------------------------------------------------------------------------------------------------------------------|
| Participantes       | Persona a monitorizar                                                                                                     |
| Descripción         | La persona a monitorizar levanta la tapa del inodoro y activa automáticamente la recolección de datos.                    |
| Condición anterior  | Ninguna                                                                                                                   |
| Flujo natural       | <ol style="list-style-type: none"> <li>1. Abrir la tapa del inodoro.</li> <li>2. Iniciar la recogida de datos.</li> </ol> |
| Flujo alternativo   | No existe flujo alternativo.                                                                                              |
| Condición posterior | Ninguna                                                                                                                   |
| Observaciones       | Ninguna                                                                                                                   |

Tabla 28: CDU-06. Levantar la tapa.

| CDU-07              | Cerrar la tapa                                                                                                                                             |
|---------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Participantes       | Persona a monitorizar                                                                                                                                      |
| Descripción         | La persona a monitorizar utiliza el inodoro y una vez finaliza cierra la tapa. Esto le indicará al sistema que ya puede finalizar la recolección de datos. |
| Condición anterior  | Levantar la tapa.                                                                                                                                          |
| Flujo natural       | <ol style="list-style-type: none"> <li>1. Cerrar la tapa del inodoro.</li> <li>2. Finalizar la recogida de datos.</li> </ol>                               |
| Flujo alternativo   | No existe flujo alternativo.                                                                                                                               |
| Condición posterior | El sistema empieza la recolección de imágenes térmicas.                                                                                                    |
| Observaciones       | Eficiencia energética                                                                                                                                      |

Tabla 29: CDU-07. Cerrar la tapa.

### 4.3. Diseño de la interacción entre todo el sistema

Como hemos podido ver en el apartado anterior, nuestro proyecto consta de multitud de sensores y placas de desarrollo. Estos elementos deberán ser capaces de interactuar entre si de una manera sencilla. A continuación, explicaremos el diseño de la interacción, y las principales razones por las cuales se ha optado por este diseño.

#### Interacción con la persona a monitorizar

Uno de los puntos más importantes en el diseño reside en la propia interacción del sistema con la persona a monitorizar. Esto es debido a que un mal diseño podría exigir a la persona realizar una serie de acciones, como pueden ser el empleo de pulsadores o botones para iniciar o finalizar la captura de datos. Este tipo de solución

puede conllevar a que la persona a monitorizar cometa errores, lo que deriva en una recogida de datos incorrecta. Es por ello, que la acción necesaria para iniciar y finalizar el sistema se basará en una acción simple y habitual, como es la apertura y cierre de la tapa. En nuestro caso, hemos utilizado un acelerómetro para determinar cuando se incrementa la aceleración en los ejes correspondientes.

## **Sincronización**

Otro punto clave del sistema se corresponde con la comunicación y sincronización entre las dos placas de desarrollo del propio dispositivo. Como hemos comentado anteriormente, por razones funcionales ha sido necesario incluir dos placas distintas para poder utilizar todos los sensores y funciones requeridas por el propio proyecto. Dado que ambas placas deben de sincronizarse para realizar la lectura de los datos, y una de ellas, en concreto la ESP8266, no dispone la capacidad de utilizar interrupciones, es necesario diseñar otro tipo de sistema que permita su correcta sincronización. Existen multitud de alternativas a la hora de sincronizar dos dispositivos tradicionales, pero a la hora de utilizar microcontroladores las opciones son más restringidas. En nuestro caso se ha optado por un sistema maestro-esclavo, que mediante una serie de cables y pulsos eléctricos en determinados pines de ambas placas conseguimos que la ESP32 active y controle el comportamiento de la ESP8266.

## **Feedback**

Otro punto muy importante a la hora de diseñar un sistema que interaccione con una persona es el feedback que le proporciona. Aunque la recogida de datos es completamente automática, requiere que la persona a monitorizar espere una cierta cantidad de tiempo antes de utilizar la cisterna, con el fin de recolectar los datos. Para proporcionarle el conocimiento a dicha persona de que el sistema ya ha finalizado la recogida de datos hemos optado por utilizar un buzzer como una alarma acústica. Utilizar este tipo de señal acústica, en lugar de algún tipo de señal lumínica, como podría ser un led, nos permite evitar crear otro módulo externo para ubicarlo en algún lugar visible por la persona a monitorizar.

## Comunicación y procesamiento de datos

Uno de los aspectos que son más relevantes en el IoT es el lugar donde se realiza el procesamiento de los datos producidos por los diferentes sensores.

### 1. Computación en el borde

En la computación en el borde [17] el procesamiento de los datos se realiza en los propios dispositivos o nodos. Los servicios basados en la nube se utilizan solamente para almacenar o monitorizar los datos. El objetivo es admitir nuevas aplicaciones con requisitos de latencia más bajos mientras se procesan los datos de manera más eficiente para ahorrar costes de red [15].

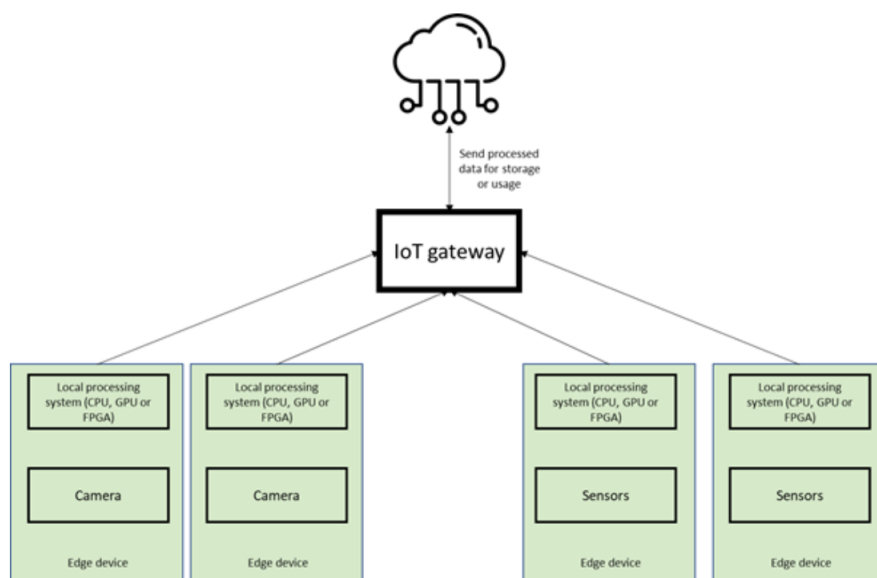


Figura 11: Esquema de funcionamiento de la computación en el borde [61].

### 2. Computación en la nube

La computación en la nube [mell2011nist] consiste en enviar la información generada por los sensores a la nube para su procesamiento o análisis. Actualmente son las plataformas como Microsoft Azure, Amazon AWS, y Google GCP las que nos brindan las principales soluciones en la nube. Dichas plataformas nos ofrecen soluciones escalables, en base a nuestras necesidades

de memoria y procesamiento. La principal deficiencia de este enfoque es que no se aprovecha la potencia de cálculo de los dispositivos nodos. Además, requerirá de un mayor ancho de banda que otras soluciones [28].

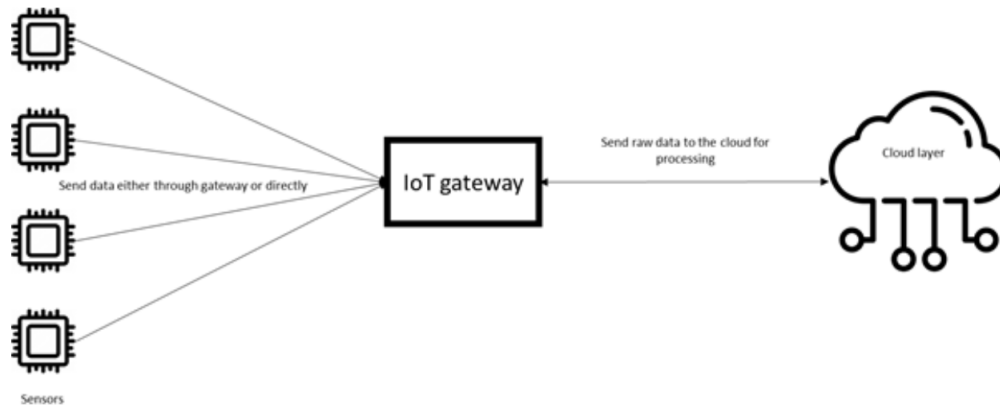


Figura 12: Esquema de funcionamiento de la computación en la nube [61].

### 3. Computación en la niebla

La computación en la niebla [18] es muy similar a la computación en el borde, intenta realizar el procesamiento de los datos lo más próximo posible a los dispositivos. El procesamiento se realiza en un nodo ubicado en la red local.

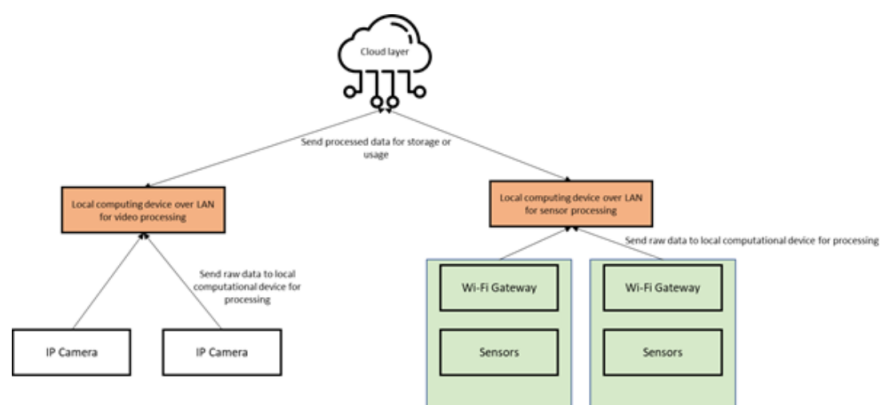


Figura 13: Esquema de funcionamiento del Fog Computing [61].

Una vez se ha iniciado el sistema y se ha realizado la sincronización entre las dos placas, llega la hora de transferir dichos datos a nuestra Raspberry Pi. Enviar los datos

recolectados por las placas de desarrollo a un dispositivo más potente como puede ser nuestra Raspberry Pi, conlleva una serie de ventajas:

- Permite realizar un tratamiento y procesamiento de los datos más agresivo. Pudiendo incluso, como veremos más adelante, generar imágenes en función de los valores de temperatura obtenidos. Cosa que sería imposible de realizar en tiempo real en un dispositivo de bajas prestaciones como pueden ser las placas esp32 y esp8266.
- Los datos se envían a un servidor ubicado en la red local. Permitiendo reducir la latencia.

Una vez que hemos visto cuales son las principales ventajas de utilizar un servidor local, procedamos a ver las opciones disponibles para establecer la conexión. Actualmente, en el mundo del IoT y de la domótica, son dos los protocolos que utilizan la mayoría de dispositivos. Estos protocolos de comunicaciones son Zigbee y Wi-Fi.

Zigbee [68] es un conjunto de protocolos con el que podemos hacer que los dispositivos que la utilicen, puedan comunicarse. Este protocolo ha estado cada vez más presente en los dispositivos comerciales al ofrecer un consumo mucho más reducido y requerir un hardware más sencillo, lo que permite abaratar los costes y permitir alargar el tiempo de funcionamiento de los dispositivos que funcionan sobre baterías.

Otra de las diferencias respecto a otros protocolos como puede ser Wi-Fi o Bluetooth reside en la estructura de la red, Figura 14. Existen tres tipos de dispositivos:

- **Controlador (ZC).** Es el dispositivo más importante de la red, solo puede haber uno, y es el que se encarga de organizar las rutas. Este dispositivo es el único que se conecta a nuestra red Wi-Fi.
- **Router (ZR).** Este dispositivo se encarga de brindar acceso a dos o más dispositivos que debido a la distancia, no es posible que se conecten directamente al controlador.

- **Dispositivo final (ZD).** Son todos los actuadores y sensores, es decir, los dispositivos que de forma activa o pasiva interactúan con nosotros.

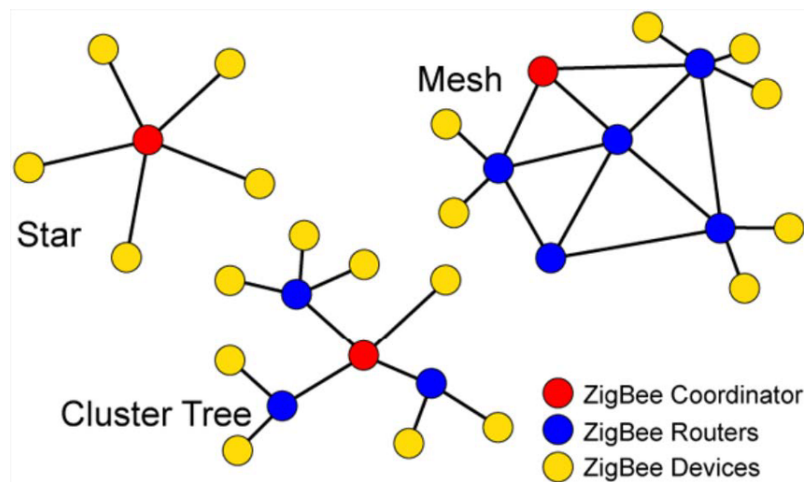


Figura 14: Estructuras de red [49].

Como se puede ver, la utilización de Zigbee puede ser muy interesante si lo comparamos con alternativas como Wi-Fi [67] + MQTT (Message Queing Telemetry Transport) [40]. No obstante, utilizar Zigbee está sometido a una serie de limitaciones. La necesidad de adquirir hardware adicional como puede ser el controlador, o dispositivos que incorporen soporte para Zigbee ha de tenerse en cuenta. Aunque Zigbee nos permita descongestionar la red Wi-Fi al crear una red diferente para los dispositivos, en nuestro caso, realizaremos un uso bajo de la red al utilizar únicamente tres dispositivos. Es por esta serie de razones que finalmente se optó por utilizar Wi-Fi y MQTT en lugar de Zigbee, pero si el objetivo es el despliegue de múltiples dispositivos en un área grande, como podría ser un hospital, se debería de optar por un dispositivo basado en Zigbee.

Dentro de nuestro sistema, las placas de desarrollo ESP32 y ESP8266 están constantemente generando datos, que son recogidos por nuestra Raspberry Pi, que actúa a modo de sumidero dentro del protocolo. Todos los datos que recibe son clasificados en función de un topic, el cual se explicará en secciones posteriores. Una vez clasificados, los datos son insertados en su correspondiente colección de la base de datos, además, antes de insertarlos, se les añade una firma temporal para determinar cuando se recibió la información. En el caso de los datos relacionados con



la cámara térmica, son enviados a una serie de funciones que se encargan de generar las imágenes térmicas, que facilitarán un análisis visual de las mismas.

#### **4.4. Diseño hardware del dispositivo**

Para construir el dispositivo vamos a necesitar una serie de sensores, y varias placas de desarrollo hardware, las cuales, se encargarán de transmitir la información a la aplicación de escritorio.

Los componentes hardware que se han utilizado se muestran a continuación.

##### **4.4.1. Placas de desarrollo**

Las placas de desarrollo son dispositivos que cuentan con un microcontrolador programable, que puede ejecutar instrucciones para un fin específico. Usualmente estas placas cuentan con entradas y salidas analógicas y/o digitales para permitir la comunicación con sensores y/o actuadores.

■ Placa de desarrollo **Lolin32 Lite**.

Esta placa de desarrollo basada en el microcontrolador ESP32, está pensada para brindar un fácil acceso a los pines del microcontrolador. Incorpora un módulo Wi-Fi que nos permitirá transmitir los datos recopilados sin la necesidad de adquirir un módulo externo, simplificando el proceso de construcción del dispositivo. También, incorpora un conversor Serie-USB, lo que permite conectar la placa directamente al ordenador para su programación.

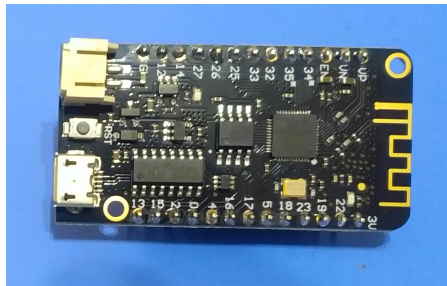


Figura 15: Placa de desarrollo Lolin32 lite.

Como características de la placa de desarrollo podemos destacar las siguientes:

- Incorpora el chip ESP32 [29], el cual integra una CPU de 32-bit Xtensa LX6 de doble núcleo a una frecuencia de 160MHz, pudiendo alcanzar los 240MHz mediante overclock. El chip cuenta con una RAM de instrucción de 64 kB y una RAM de datos de 96 kB.
- Memoria Flash de 4MB.
- Wi-Fi: 802.11 b/g/n.
- Bluetooth: v4.2 BR/EDR y BLE.
- Acceso a los pines del microcontrolador.
- Led de estado: Proporciona información relativa al funcionamiento del ESP32 y la carga de la batería.
- Conversor Serie-USB: Permite alimentar la placa tanto como programarla.
- Botón de Reset.
- Puerto de baterías: permite conectar baterías con una tensión máxima de 3.7V.

- Perfiles energéticos: incluye diferentes modos de funcionamiento, que serán útiles para reducir el consumo energético.

Lo más adecuado para trabajar con una placa de desarrollo es conectarla a una placa de prototipado. Esto nos dará un mayor estabilidad y nos permitirá realizar las conexiones de una manera más cómoda.

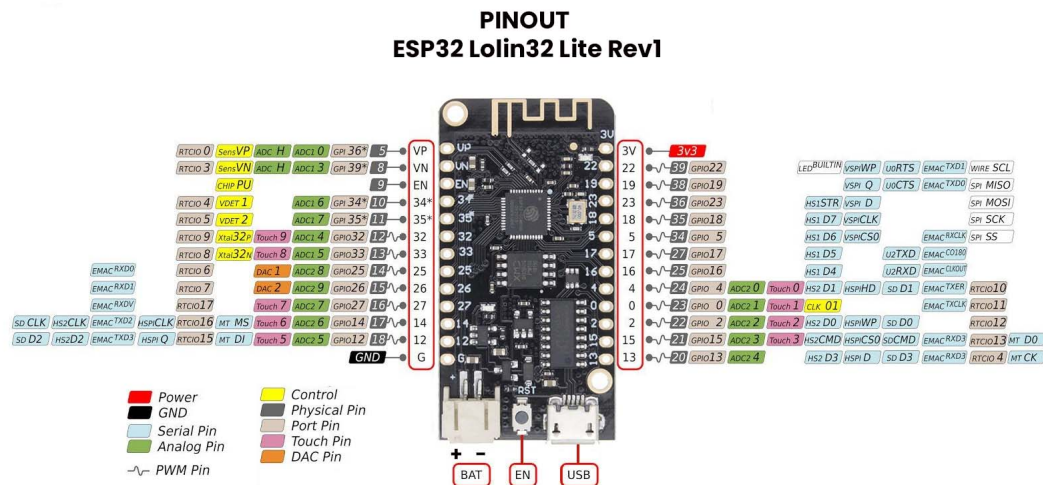


Figura 16: Pines Lolin32 lite.

Como se puede observar en la Figura 16, esta placa de desarrollo ofrece una amplia variedad de pines con multitud de funciones asociadas, aunque los más utilizados en este proyecto son:

- Pines GND y 3V3, los cuales se usan para suministrar energía a los sensores.
- Pines digitales, se usan tanto para recibir los datos de los sensores como para el manejo de interrupciones.

Hay que tener en cuenta que algunos pines de la placa de desarrollo son usados internamente por el microcontrolador para la comunicación con las memorias, mientras que otros pueden producir señales digitales al encenderse. Por eso es recomendable no utilizar estos pines, con el fin de evitar falsas lecturas y errores de funcionamiento.

Por tanto, las principales razones por las que hemos elegido el kit de desarrollo Lolin32 lite como una de las plataformas hardware para nuestro proyecto son, la integración de un módulo Wi-Fi, que nos permite transmitir los datos sin la necesidad de adquirir hardware adicional y la disponibilidad de funciones avanzadas como el modo Deep Sleep.

■ Placa de desarrollo **Placa de desarrollo NodeMcu V3.**

Esta placa de desarrollo es la predecesora de la ESP32. Hemos incorporado esta placa de desarrollo porque podemos obtener 5V desde uno de sus pines.

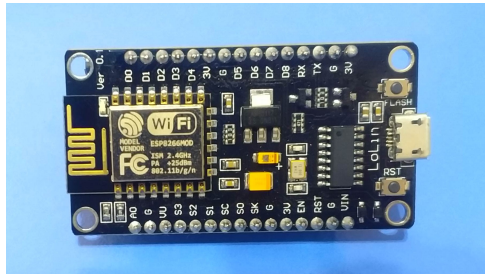


Figura 17: Placa de desarrollo NodeMcu V3.

Entre las principales características del kit de desarrollo podemos destacar las siguientes:

- Incorpora el módulo ESP12E, basado en el chip ESP8266 [30], el cual integra una CPU RISC de 32-bit Tensilica Xtensa LX106 a una frecuencia de 80MHz, pudiendo alcanzar los 160MHz mediante overclock. El chip cuenta con una RAM de instrucción de 64 kB y una RAM de datos de 96 kB.
- Memoria Flash de 4MB.
- Wi-Fi: 802.11 b/g/n.
- Bluetooth: v4.2 BR/EDR y BLE.
- Acceso a los pines del microcontrolador.
- Led de estado. Proporciona información relativa al funcionamiento del ESP32 y la carga de la batería.
- Conversor Serie-USB: Permite alimentar la placa tanto como programarla.
- Botón de Reset y Flash.

■ Placa de desarrollo **Raspberry Pi 3 Model B+**

Esta placa de desarrollo es una de las más populares en el mundo maker. Nos ofrece mucha más capacidad de cómputo que las placas ESP32 y ESP8266. Además, será posible ejecutar un sistema operativo de escritorio, permitiendo al usuario/cuidador visualizar los datos recopilados por los dispositivos.



Figura 18: Placa de desarrollo Raspberry Pi 3 Model B+ [48].

Entre las principales características de esta placa de desarrollo podemos destacar las siguientes:

- Incorpora un procesador Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC @ 1.4GHz.
- 1GB LPDDR2 SDRAM.
- 2.4GHz and 5GHz IEEE 802.11.b/g/n/ac wireless LAN, Bluetooth 4.2, BLE.
- 4 puertos USB 2.0.
- Puerto ethernet.

Hemos incorporado esta placa al proyecto para ubicar los diferentes sistemas que se encargarán de gestionar el flujo de datos entre las placas ESP8266 Y ESP32, con la base de datos.

#### 4.4.2. Sensores

- **Adafruit AMG8833** cámara térmica IR.

Este dispositivo integra el sensor AMG8833 Grid-EYE de Panasonic, el cual está formado por una matriz 8x8 de sensores térmicos infrarrojos. Este sensor cuenta con un rango de medición de 0°C a 80 °C con una precisión de  $\pm 2^{\circ}\text{C}$ .

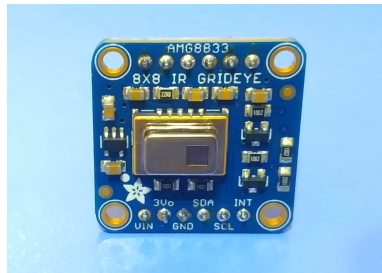


Figura 19: Sensor Adafruit AMG8833.

Sus principales características son las siguientes:

- Rango de medición de 0°C a 80 °C con una precisión de  $\pm 2^{\circ}\text{C}$ .
- La comunicación se realiza mediante I2C. Para esta finalidad incorpora dos pines, uno denominado SDA y otro SCL.
- Incorpora un regulador de voltaje y un conversor, permitiendo una tensión máxima entre 3,3 y 5 Voltios, lo cual nos permitirá alimentarlo directamente desde nuestras placas de desarrollo.
- 10Hz de tasa de refresco, podremos obtener hasta 10 imágenes por segundo.
- Incorpora un pin de interrupción. Podremos programar el sensor para que una vez sea superado cierto umbral de temperatura, enviar una señal a la placa de desarrollo.

Una vez vistas las principales características de este sensor podemos proceder a explicar las razones de su elección para este proyecto.

Actualmente este tipo de dispositivos para la captura de imágenes infrarrojas, tienen un coste que en ciertas situaciones puede resultar poco accesible. Entre los principales limitantes de este dispositivo se encuentran la escasa resolución que nos ofrece, 8x8 píxeles, para un total de 64 píxeles. Esta resolución en ciertos ámbitos o aplicaciones podría suponer un serio limitante a la hora de implementar este dispositivo. Pero dada la naturaleza de las imágenes que debemos capturar, nos proporciona un componente anónimo. Además, el precio se ve recudido de manera notable en base a la resolución.

■ **Acelerómetro MMA8451.**

El MMA8451 es un acelerómetro digital que nos permite detectar movimiento, inclinación y la orientación, aunque esta última de una manera aproximada.

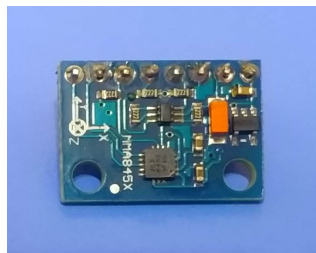


Figura 20: Acelerómetro MMA8451.

Sus características son las siguientes:

- Incorpora un ADC de 14 bits, lo que nos proporciona una mayor resolución.
- Detección de la aceleración en los ejes X,Y,Z.
- Consumo de energía reducido.

Las principales razones que han llevado a la inclusión de este sensor en el proyecto son las siguientes:

- Determinar cuando la persona a monitorizar se dispone a utilizar el inodoro.
- Determinar el momento en el que inicia la acción, su finalización y el tiempo transcurrido.

- Sensor de gas **MQ2**.

Este dispositivo, Figura 21, nos permite medir las concentraciones de diferentes tipos de gases presentes en el aire. Es un MOS (Metal Oxide Semiconductor), ya que su capacidad de detección está basada en los cambios que se producen en la resistencia del material reactivo cuando los gases entran en contacto con él. Puede detectar Gas Licuado de Petroleo (GLP), humo, alcohol, propano, hidrógeno, metano y monóxido de carbono en concentraciones de 300ppm a 10000ppm.

Como principales desventajas encontramos su consumo energético, el cual es de unos 150mAh. Este consumo podría parecer reducido, pero al operar sobre baterías, se convierte en un factor a tener en cuenta. Además, el sensor necesita un tiempo de precalentado para poder realizar mediciones precisas, imposibilitando la opción de regular el consumo energético mediante el uso de transistores.

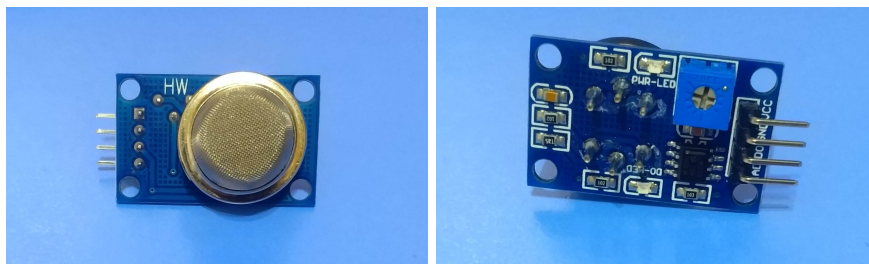


Figura 21: Sensor de gas MQ2.

Como se puede observar en la Figura 22, este sensor viene acompañado de una gráfica que relaciona el valor de  $R_s/R_0$  con el número de ppm presentes para diferentes gases. Gracias a esta gráfica podremos calcular el valor aproximado de partículas en función del valor de la resistencia del sensor de gas.



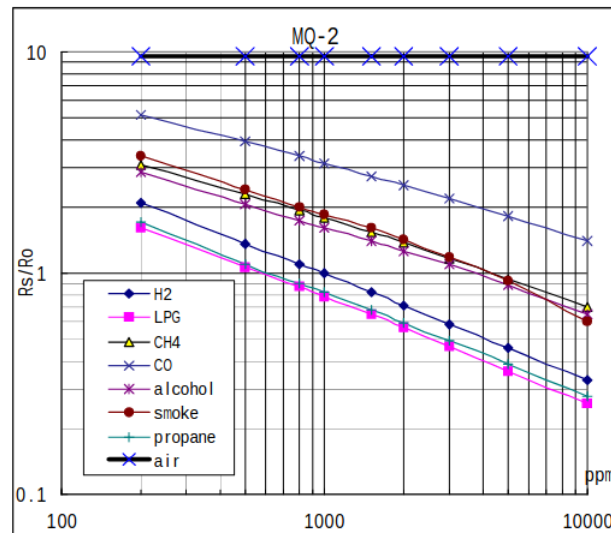


Figura 22: Sensibilidad del sensor MQ2 en función del gas [39].

Para poder realizar la lectura de los datos será necesario emplear un divisor resistivo. Esto es debido a que la salida analógica del sensor proporciona un valor entre 0 y 5 V, pero nuestra placa sólo soporta un valor máximo de entrada en los pines de 3,3V.

Para ello necesitaremos la *fórmula general de la ley de Ohm*, la cual establece que:

$$V = R * I$$

donde V es la diferencia de potencial, I la intensidad de la corriente y R la resistencia eléctrica. Las unidades de esas tres magnitudes en el sistema internacional de unidades son, respectivamente, voltios (V), ohmios ( $\Omega$ ) y amperios (A).

$$V_{in} = (R_1 + R_2) * I$$

Si despejamos la intensidad obtenemos:

$$I = \frac{V_{in}}{(R_1 + R_2)}$$

$$V_{out} = R_2 * I \Rightarrow V_{out} = R_2 * \frac{V_{in}}{(R_1 + R_2)}$$

Una vez tenemos la fórmula, solo es necesario asignar un valor inicial a  $R_1$  o a  $R_2$ . En nuestro caso le hemos asignado a  $R_1$  un valor de 10 kOhm.

$$R_1 = 10 \text{ kOhm}, V_{out} = 3,3V, V_{in} = 5V$$

$$3,3 = R_2 * \frac{5}{(10k + R_2)}$$

$$3,3 * R_2 + 10k * 3,3 = 5 * R_2$$

$$R_2 = \frac{-10k * 3,3}{3,3 - 5} = \frac{10k * 3,3}{5 - 3,3} = 20k$$

Por tanto:

$$R_2 = 20 \text{ kOhm}$$

Una vez hemos calculado todos los valores ya podemos construir el divisor resistivo conforme aparece en la Figura 23. Nótese que podemos sustituir las resistencias  $R_1$  o  $R_2$  por una combinación en serie de tal manera que sus valores sumen el valor de  $R_1$  o  $R_2$ .

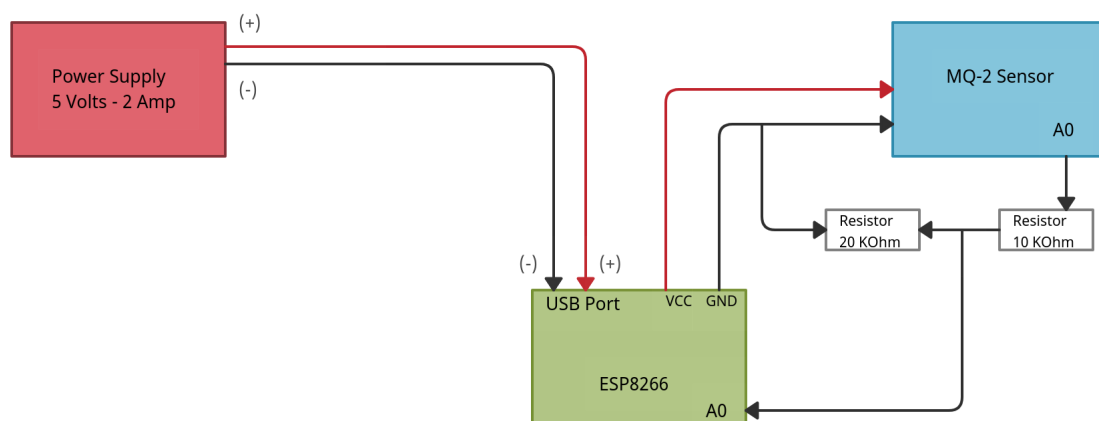


Figura 23: Divisor resistivo.

Mediante la adición de este sensor al proyecto, esperamos poder obtener lecturas de los valores de los gases presentes en el ambiente, las cuales nos permitan monitorizar los gases. Hay que recordar que el uso de este sensor para la finalidad de este proyecto no ha sido prevista por el fabricante del dispositivo, y que no existe ningún trabajo previo sobre su utilización en este ámbito. Esto quiere decir que el sensor podría comportarse de manera anómala, arrojar valores no concluyentes o incluso no detectar variaciones significativas.

- Sensor de gas **MQ135**.

Este sensor, al igual que el MQ2, está diseñado para realizar lecturas de los gases presentes en el ambiente. El sensor MQ-135 es ideal para la detección de gas amoníaco, benceno y alcohol, aunque también es sensible al humo y a otros gases tóxicos, Figura 25. El sensor puede detectar concentraciones de gas entre 10 y 1000 ppm, dependiendo del tipo de gas.

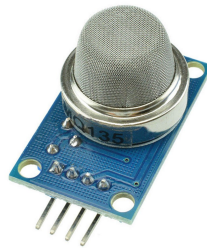


Figura 24: Sensor MQ135.

Dado que el funcionamiento de este sensor es equivalente al anterior, los esquemas de conexiones y el divisor resistivo, pueden ser reutilizados para este sensor.

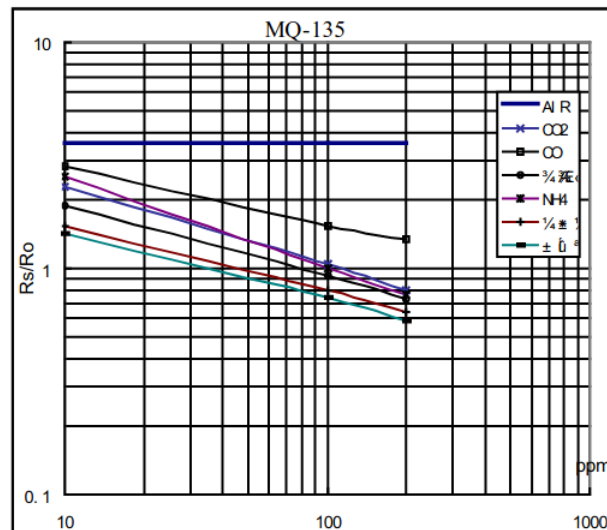


Figura 25: Sensibilidad del sensor MQ135 en función del gas [60].

Debido a la ubicación del sensor, deberemos tener en cuenta los valores de humedad y temperatura. Esto se debe a que el comportamiento del sensor se ve afectado en función de dichos valores, Figura 26.

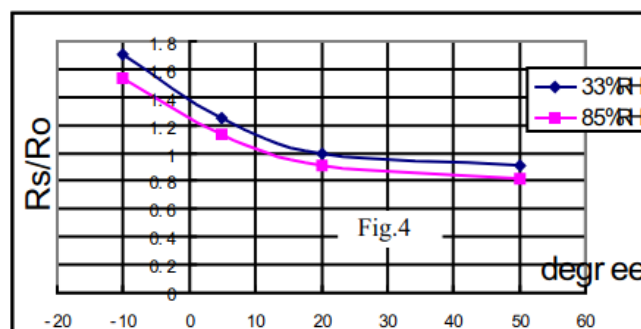


Figura 26: Dependencia del MQ-135 por temperatura y humedad;  $R_o$ = Resistencia del sensor a 100 ppm de  $NH_3$  en aire a 33 % RH y 20 C;  $R_s$ = Resistencia del sensor con 100 ppm de  $NH_3$  en el aire a diferentes temperaturas y humedades [60].

Al igual que el sensor MQ2, este sensor necesita de un tiempo de precalentado para poder obtener valores fiables.

Hemos incluido este sensor en el proyecto para la recogida inicial de muestras. Posteriormente realizaremos una comparación de estos valores con los recogidos por el sensor MQ2.

#### 4.4.3. Actuadores

- **Buzzer pasivo.**

Los buzzer o altavoces son dispositivos que nos permiten transformar señales eléctricas en ondas de sonido. La principal diferencia entre un buzzer pasivo y uno activo reside en la capacidad de modular la onda de sonido resultante.

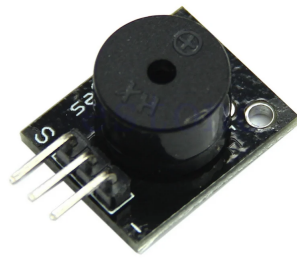


Figura 27: Buzzer pasivo.

Usaremos este dispositivo para avisar acústicamente a la persona a monitorizar de que se ha finalizado la recogida de datos, y por tanto ya puede proceder a accionar la cisterna.

#### 4.4.4. Iteraciones realizadas

Una vez hemos presentado los distintos componentes que contendrá nuestro prototipo de producto IoT, ya podemos empezar a construirlo. Debido al enfoque de desarrollo basado en iteraciones, nuestro prototipo ha estado en constante evolución, ya sea tanto para incluir mejoras o para proporcionar un dispositivo funcional en cada iteración.

##### **Primera Iteración**

En esta primera iteración se construyó un primer prototipo [28], que estaba formado por el sensor de gas y la placa ESP8266. Los componentes se colocan en una protoboard para facilitar su conexión. Este prototipo tenía la finalidad de comprobar

el correcto funcionamiento de la recogida de gases, su envío a la Raspberry Pi a través de MQTT, y su posterior inserción a la base de datos.

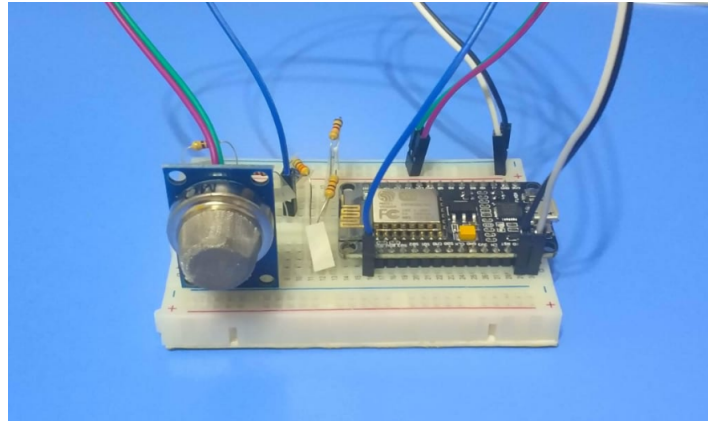


Figura 28: Primer prototipo.

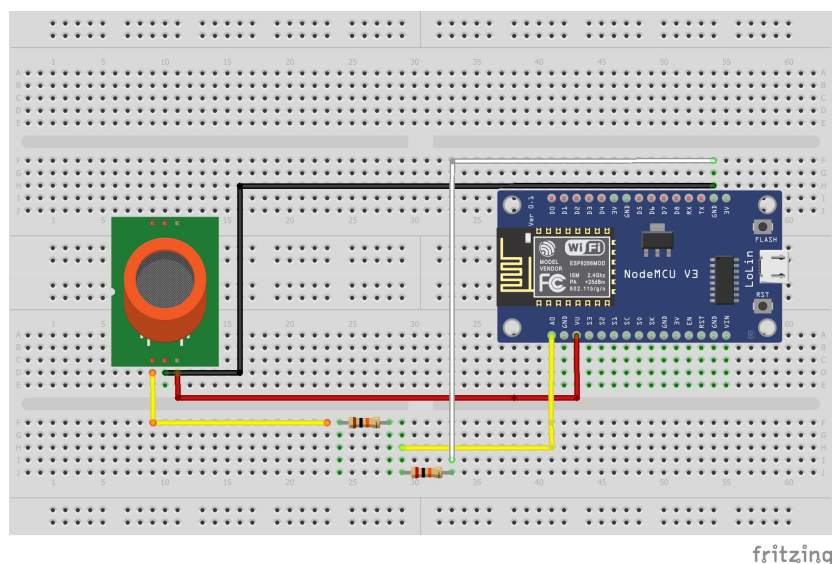


Figura 29: Esquema de conexiones de la primera iteración.

Como podemos ver, se trata de un dispositivo de tamaño reducido y que no incorpora todas las funcionalidades del proyecto, pero nos permite validar nuestro avance mostrando un producto funcional.

## Segunda iteración

En la segunda iteración se construye un dispositivo, Figura 30, que integra prácticamente la totalidad de los componentes. Este prototipo es capaz de realizar

todas las recogidas de datos demandadas por el proyecto, pero durante la fase de pruebas se detectó un problema con la placa base ESP8266. Esta placa, de menor tamaño que el modelo utilizado en la versión final, presentaba un problema en el suministro eléctrico al sensor de gas. Esto resultó en la lectura incorrecta de los valores, lo que propició su sustitución por otro modelo de mayores dimensiones.

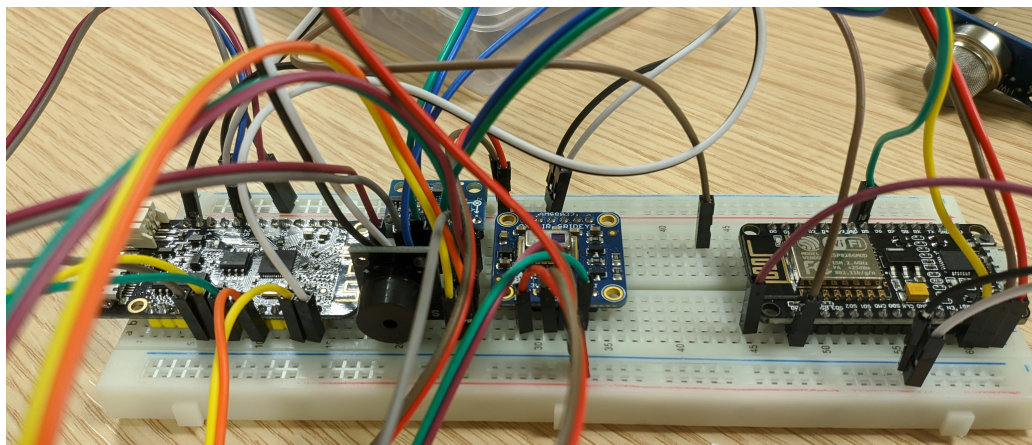


Figura 30: Segundo prototipo.

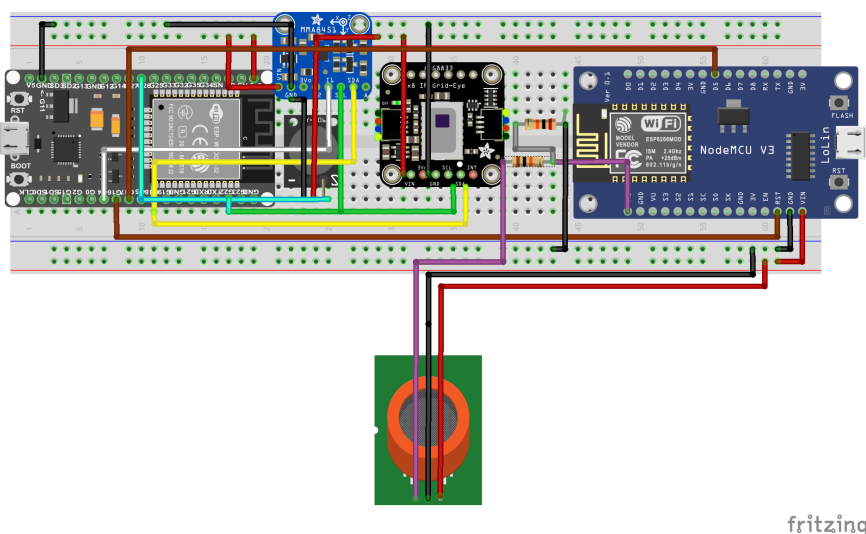


Figura 31: Esquema de conexiones de la segunda iteración.

### Tercera iteración

En la tercera y última iteración realizada, los componentes se distribuyeron en dos breadboard previamente unidas. Este cambio nos permite conectar directamente la nueva placa ESP8266, que al ser de mayores dimensiones que el modelo



anterior, no permitía su colocación en una única breadboard. Además, se procedió a fijar los componentes mediante pegamento termofusible, para evitar desconexiones accidentales.

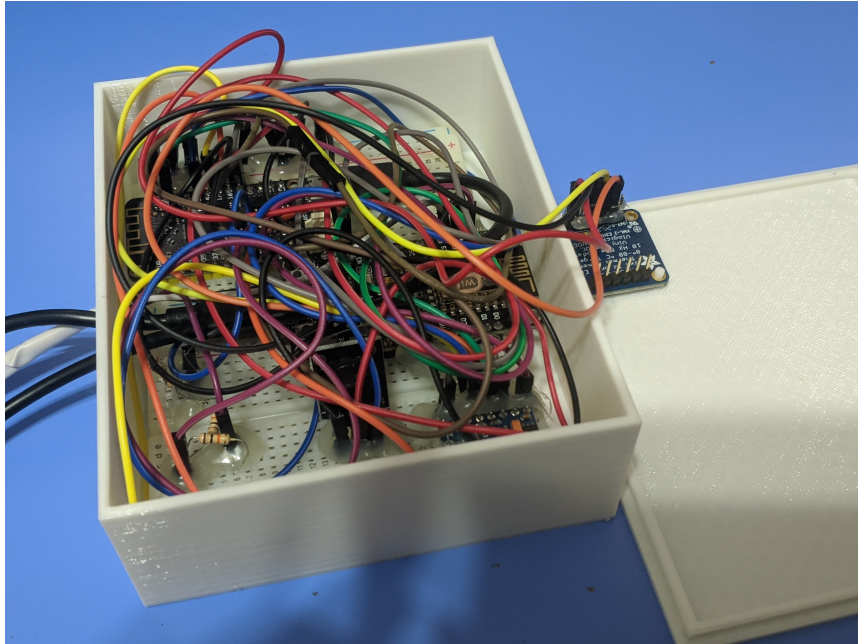


Figura 32: Tercer prototipo.

Como se puede ver en la Figura 32, el hardware ya se encuentra ubicado dentro de la carcasa diseñada en 3D.

#### **4.5. Diseño físico del dispositivo**

Para facilitar la integración de nuestro dispositivo en un entorno real y de cara a un prototipo final comercial hemos diseñado y construido unas carcasas. Estas carcasas nos permitirán fijar nuestro dispositivo de manera más óptima a las tapas de los inodoros y adicionalmente nos permitirán proteger los componentes internos de nuestro dispositivo. Para realizar los diseños hemos utilizado el software FreeCAD [58].

FreeCAD es una aplicación libre de diseño asistido por computadora (CAD) en tres dimensiones que nos permite diseñar modelos y exportarlos para su posterior impresión mediante una impresora 3D.



Entre los diferentes factores que han contribuido a tomar la decisión de utilizar este software se encuentran los siguientes:

- FreeCAD es completamente gratuito.
- La curva de aprendizaje es muy corta y, además, cuenta con una gran comunidad con la que podremos resolver nuestras dudas.
- Permite exportar nuestros diseños a una gran variedad de formatos, entre los que se encuentra STL (Standard Tessellation Language), que es el utilizado por las herramientas de impresión en 3D.
- Existen millones de modelos predefinidos que podremos utilizar para agilizar el diseño de nuestros propios modelos.

#### **4.5.1. Iteraciones del modelo 3D**

Como hemos comentado anteriormente, nuestra metodología de trabajo está basada en iteraciones o sprints. Es por esta razón, que durante las diferentes iteraciones realizadas, el diseño del modelo 3D ha estado sometido a cambios.

##### **Primera iteración**

La primera iteración del modelo, Figura 33, consistía en un único módulo que albergaría toda la electrónica del dispositivo IoT. Esta primera iteración contemplaba el uso de baterías internas, por lo que no era necesario realizar aberturas adicionales para conectar los elementos a una fuente de alimentación externa, permitiendo aislar mejor al dispositivo. Para fijar los distintos componentes internos en una posición fija se optó por utilizar un sistema de columnas, que se insertan en los propios orificios existentes de los componentes. Esto, junto a la aplicación de un pegamento termofusible, evita que los componentes se desplacen libremente por el interior de la carcasa.

Las principales motivos por los que este diseño se descartó son los siguientes:

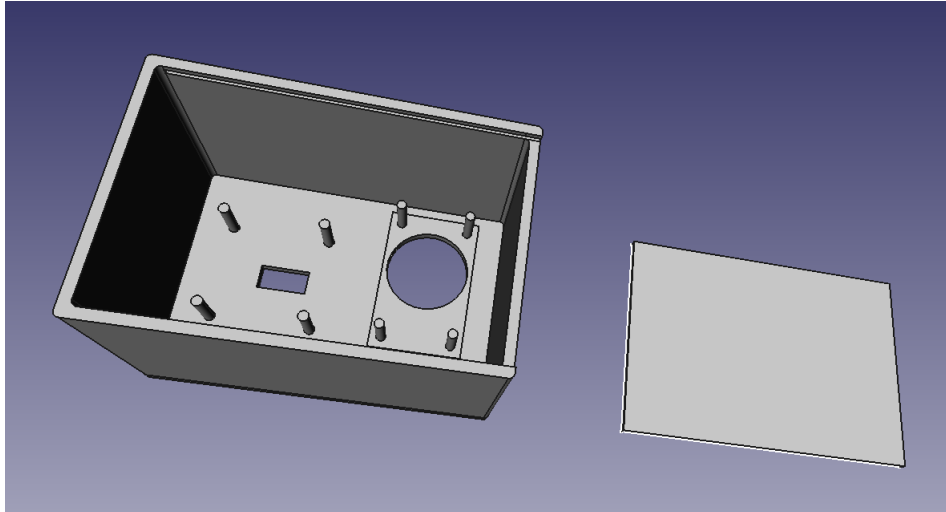


Figura 33: Modelo original.

- Como resultado de apilar los diferentes componentes de forma vertical, la altura final del dispositivo dificultaba su integración en un entorno real.
- El sistema de carriles implementado para la apertura y cierre de la tapa era susceptible de producir errores de impresión.

### Segunda iteración

En la segunda iteración, se tomó la decisión de dividir los componentes en dos módulos:

- Módulo principal 34. Este módulo contendría toda la electrónica del dispositivo IoT a excepción del sensor de gas. Al igual que la anterior interacción, este módulo se fijaría en la tapa del inodoro para recoger los datos una vez que la persona a monitorizar finalice su uso.
- Módulo del sensor de gas 35. Este módulo contendría exclusivamente al sensor de gas. Esto nos permitiría ubicar el módulo, dependiendo del modelo del inodoro, en una posición más optima para la recogida de los gases, mientras la persona a monitorizar realiza un uso del mismo.

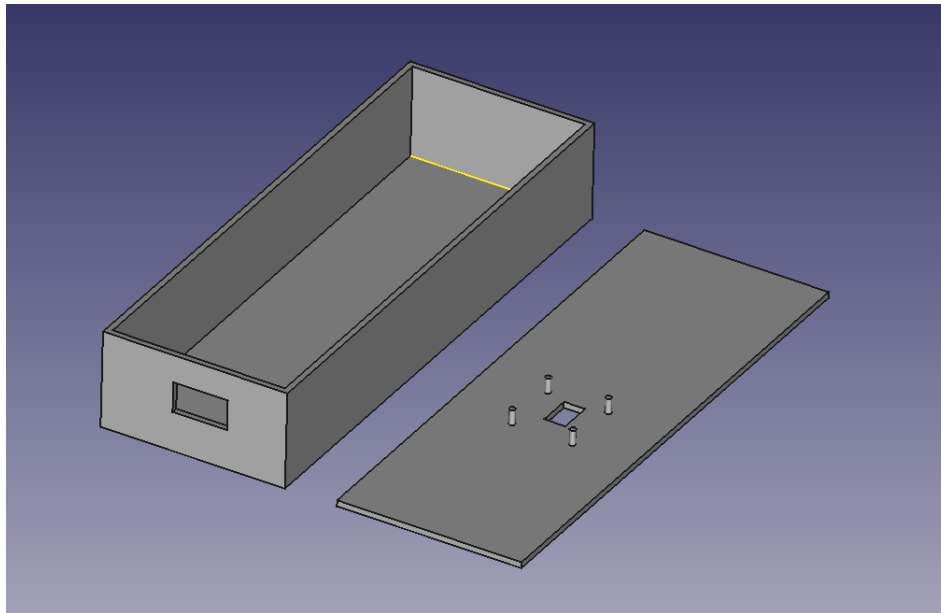


Figura 34: Modelo 3D del módulo principal.

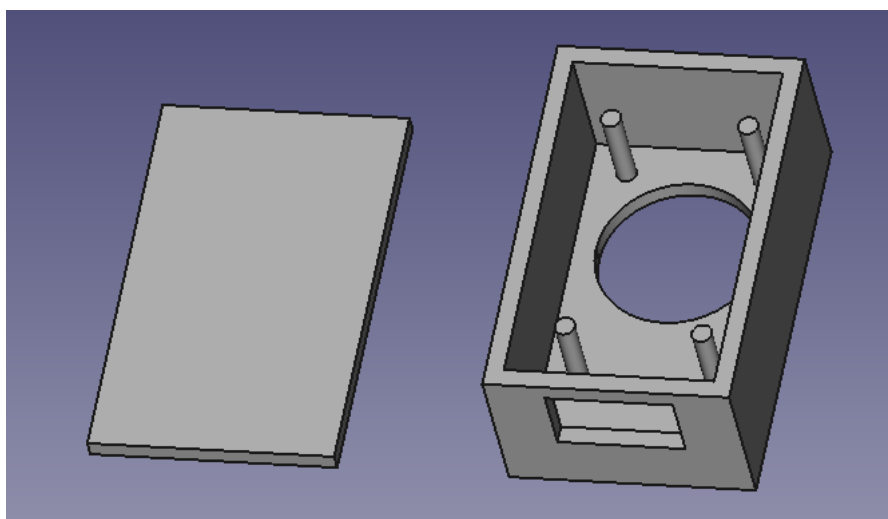


Figura 35: Modelo 3D del módulo para el sensor de gas.

Como podemos ver, los principales cambios aparte de la división en dos módulos se centran en aspectos como el sistema de fijación de la tapa o la disposición de los componentes electrónicos en su interior. El sistema de fijación basado en carriles ha sido sustituido por un sistema que mediante la aplicación de resinas, pegamento o cintas, fijan la tapa directamente al cuerpo del modelo.

Los principales motivos que llevaron a realizar un posterior diseño de la pieza son los siguientes:

- Como se ha comentado anteriormente en la sección de diseño hardware del dispositivo, un problema derivado con el tamaño de la placa de desarrollo propició el cambio de ubicar los componentes en una nueva base.
- Debido al sistema de fijación empleado, resultaba complicado alinear las diferentes partes para realizar la fijación de las mismas.

### Tercera iteración

En la tercera y última iteración, se diseñó el modelo del módulo principal, Figura 36, para contener la nueva distribución de hardware previamente vista, Figura 32.

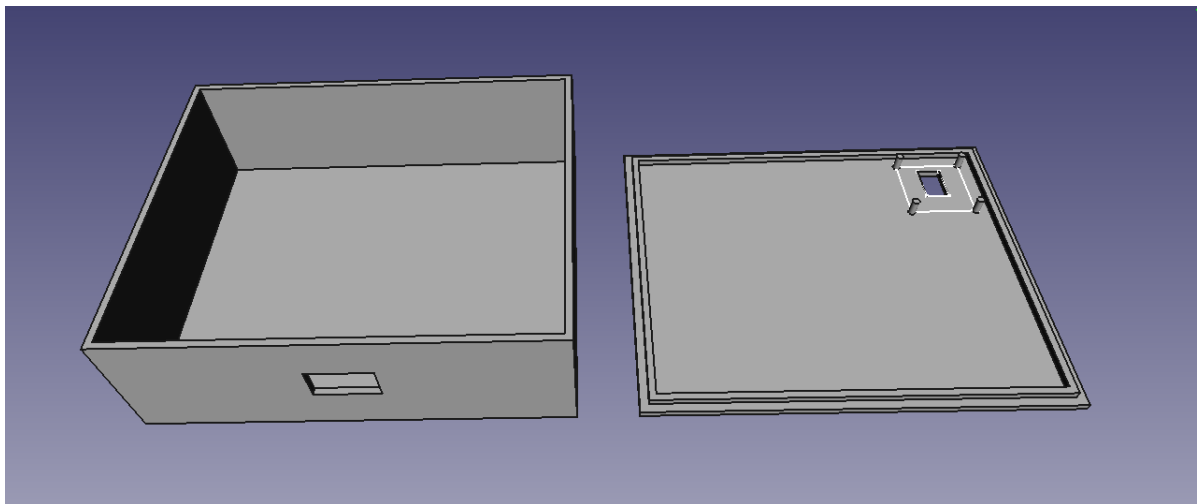


Figura 36: Modelo 3D del módulo principal.

En este diseño, se incorpora un saliente en la tapa de la carcasa, permitiendo fijar de una manera más sólida ambas partes.

Una vez hemos finalizado ambos diseños, generamos los correspondientes archivos en formato STL. Estos archivos serán los que carguemos en el software de impresión Ultimaker Cura [3]. Una vez carguemos los archivos, nos mostrará las diferentes capas necesarias para completar la pieza, además de una estimación del tiempo necesario para su impresión. Las piezas finales, Figura 38, se imprimieron utilizando filamento PLA.

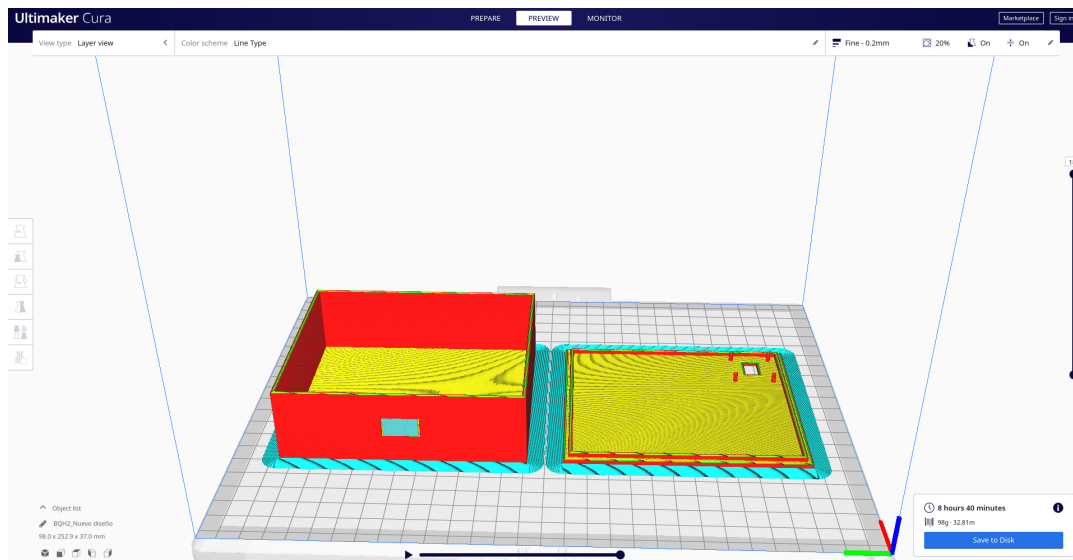


Figura 37: Pieza en el software de impresión Ultimaker Cura [3].

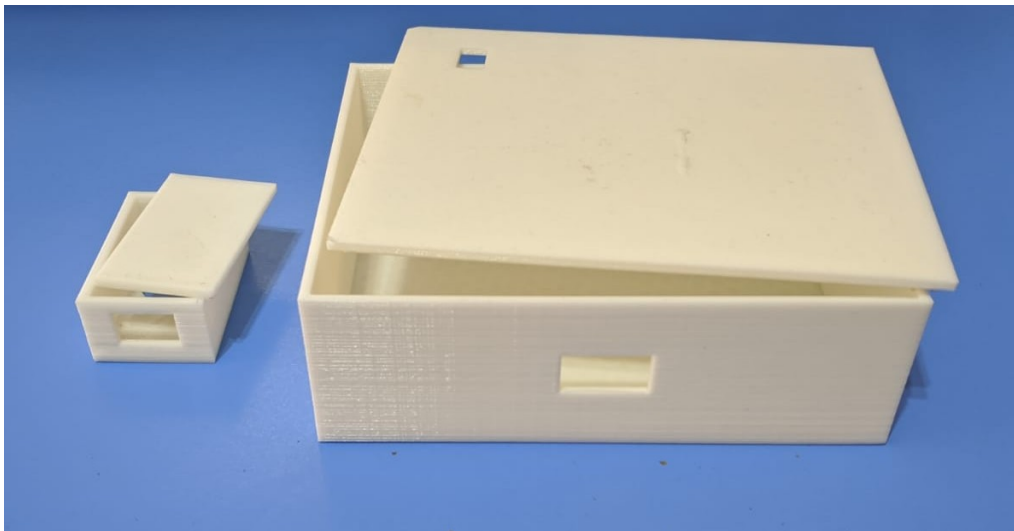


Figura 38: Piezas impresas en 3D.

## 4.6. Diseño del modelo de datos

En esta sección se valorarán los factores que han llevado a seleccionar el tipo de base de datos, atendiendo al volumen de datos y a los diferentes tipos que se van a necesitar.

Para establecer la estructura de la base de datos se utilizará el método del modelado entidad - relación. Para poder aplicar este método, será necesario identificar las distintas entidades intervinientes en la aplicación.

#### **4.6.1. Entidades del modelo de datos**

En este caso las entidades serán todas aquellas que se refieran a los objetos propios de la aplicación. Dado la naturaleza de nuestra aplicación, los datos almacenados en la base de datos se corresponderán con el propio contenido procedente de los sensores. Es decir, existirá una entidad para los datos termales, otra entidad para los datos relacionados con la apertura de la tapa, otra entidad que se corresponderá con los valores obtenidos por el sensor de gas, y dos entidades que contendrán los resultados de la prueba de gases con el sensor MQ135 y con el MQ2. Las operaciones que se realizarán serán básicas, es decir, de tipo CRUD (inserción, modificación, borrado y selección).

#### **4.6.2. Justificación de la elección de MongoDB**

Una vez que se tienen claros los diferentes elementos a almacenar en nuestra base de datos por las distintas entidades, se realizará una pequeña revisión de los principales beneficios y carencias de los dos tipos de bases de datos más empleadas, SQL Y NoSQL.

##### **Base de datos de tipo SQL**

Son un tipo de bases de datos que garantizan siempre las propiedades de atomicidad mediante la aplicación de rollback. Además, los datos deben cumplir requisitos de integridad tanto en tipo de dato como en compatibilidad. Sin embargo, la atomicidad de las propias operaciones penaliza el rendimiento de la propia base de datos. Su escalabilidad también es más limitada en comparación con bases de datos de tipo NoSQL.

##### **Bases de datos de tipo NoSQL**

Las bases de datos de tipo NoSQL destacan por la escalabilidad y su carácter descentralizado. Permitiendo soportar estructuras distribuidas. Suelen ser bases de datos mucho más abiertas y flexibles. Permiten adaptarse a necesidades de proyectos mucho más fácilmente que los modelos de Entidad Relación. Se pueden ejecutar en

máquinas con pocos recursos. Realizan optimizaciones de las consultas. Sin embargo, este tipo de bases de datos no contemplan la atomicidad y la integridad de los datos.

Como podemos ver, en base a nuestros requisitos de la base de datos, podemos determinar que la opción más interesante es el uso de una base de datos de tipo NoSQL frente a una base de datos de tipo SQL.

En el mercado existen una amplia variedad de bases de datos de tipo NoSQL. Entre las cuales se encuentran:

- SimpleDB: Tiene alta disponibilidad y flexibilidad, debido a que administra automáticamente múltiples copias de los datos. Los desarrolladores almacenan los datos y los consultan mediante solicitudes de servicios web.
- Cassandra: Esta base de datos está basada en un modelo wide-column, donde el formato y nombre de las columnas puede variar de línea en línea en la misma tabla. Permite distribuir grandes cantidades de datos entre distintas instancias. La arquitectura de Cassandra está basada en una serie de nodos iguales, que se comunican con un protocolo P2P.
- MongoDB: Guarda estructuras de datos BSON con un esquema dinámico, haciendo que la integración de los datos para ciertas aplicaciones sea más fácil y rápida.

A continuación se presenta una tabla que compara diferentes aspectos de algunas de las bases de datos NoSQL que existen.

| Base de datos         | Tipo        | ACID | Casos de uso                                                                     | Seguridad                                                                       |
|-----------------------|-------------|------|----------------------------------------------------------------------------------|---------------------------------------------------------------------------------|
| Cassandra             | Wide-column | No   | Análisis en tiempo real, análisis social                                         | Incluye seguridad para la autenticación, encriptación, auditoría y autorización |
| Neo4j                 | Graph       | Sí   | IA (Inteligencia Artificial), servicios de recomendación y protección de fraudes | Dispone de autorización, roles y encriptación                                   |
| Google Cloud Bigtable | Wide-column | No   | Gestión de datos IoT y datos de series de tiempo                                 | Seguridad proporcionada por el proveedor                                        |

Tabla 30: Comparación de bases de datos NoSQL.



| Base de datos | Tipo           | ACID | Casos de uso                                                     | Seguridad                                                               |
|---------------|----------------|------|------------------------------------------------------------------|-------------------------------------------------------------------------|
| Redis         | Key-value      | Sí   | Cacheado, filtrado, gestión de colas y estadísticas              | Protección activada por defecto y ofrece sugerencias sobre la seguridad |
| MongoDB       | Document-based | Sí   | Gestión de datos IoT, desarrollo de apps y gestión de inventario | Dispone de autenticación, autorización y encriptación                   |

Tabla 31: Comparación de bases de datos NoSQL.

La selección de MongoDB ha estado propiciada por una serie de razones. MongoDB permite la creación de índices para acelerar las búsquedas, lo que resulta de utilidad cuando el volumen de datos se incrementa notablemente. Al estar enfocada al IoT cuenta con una serie de herramientas que permiten realizar un análisis automático de los datos, aunque para acceder a dichas funciones es necesario adquirir una versión comercial. El modelo de datos basado en documentos nos permite definir estructuras que agrupen diferentes campos.

#### 4.6.3. Colecciones de datos

Como se ha determinado anteriormente, nuestra base de datos dispondrá únicamente de cuatro entidades, las cuales no se relacionan entre si. A continuación, se procederá a determinar las propiedades que han de poseer para el adecuado funcionamiento de nuestra aplicación.

| THERMAL_DATA  | Descripción                                             | Tipo de variable |
|---------------|---------------------------------------------------------|------------------|
| id            | Identificador único                                     | ObjectId         |
| sensor        | Identificador del sensor                                | String           |
| thermal_array | Lista de valores de temperatura recogidos por el sensor | Array            |
| img_id        | Número de secuencia de la imagen                        | Integer          |
| control_data  | Indica si el dato es una muestra de control             | Boolean          |
| date          | Fecha y hora en formato internacional                   | ISODate          |

Tabla 32: Colección de datos para la entidad THERMAL\_DATA.

| GAS_DATA | Descripción                           | Tipo de variable |
|----------|---------------------------------------|------------------|
| id       | Identificador único                   | ObjectId         |
| sensor   | Identificador del sensor              | String           |
| raw_adc  | Voltaje generado por el sensor de gas | Integer          |
| ppm      | Partículas por millón detectadas      | Integer          |
| date     | Fecha y hora en formato internacional | ISODate          |

Tabla 33: Colección de datos para la entidad GAS\_DATA.

| USE_DATA   | Descripción                                     | Tipo de variable |
|------------|-------------------------------------------------|------------------|
| id         | Identificador único                             | ObjectId         |
| sensor     | Identificador del sensor                        | String           |
| toilet_lid | Indica si la tapa se ha abierto o cerrado       | String           |
| minutes    | Número de minutos que la tapa ha estado abierta | Integer          |
| seconds    | Número de segundos restantes                    | Integer          |
| date       | Fecha y hora en formato internacional           | ISODate          |

Tabla 34: Colección de datos para la entidad USE\_DATA.

| GAS_TEST       | Descripción                           | Tipo de variable |
|----------------|---------------------------------------|------------------|
| id             | Identificador único                   | ObjectId         |
| Sensor         | Identificador del sensor              | String           |
| Type_id        | Tipo de sensor                        | String           |
| Sequence_id_id | Número de secuencia de la muestra     | Integer          |
| Alcohol_id     | PPM del gas                           | Double           |
| H2             | PPM del gas                           | Double           |
| C0             | PPM del gas                           | Double           |
| LPG            | PPM del gas                           | Double           |
| Propane        | PPM del gas                           | Double           |
| date           | Fecha y hora en formato internacional | ISODate          |

Tabla 35: Colección de datos para la entidad GAS\_TEST. PPM: Partes por millón.

| GAS_TEST        | Descripción                           | Tipo de variable |
|-----------------|---------------------------------------|------------------|
| id              | Identificador único                   | ObjectId         |
| Sensor          | Identificador del sensor              | String           |
| Type_lid        | Tipo de sensor                        | String           |
| Sequence_id_lid | Número de secuencia de la muestra     | Integer          |
| Alcohol_id      | PPM del gas                           | Double           |
| CO2             | PPM del gas                           | Double           |
| CO              | PPM del gas                           | Double           |
| Tolueno         | PPM del gas                           | Double           |
| NH4             | PPM del gas                           | Double           |
| Acetona         | PPM del gas                           | Double           |
| date            | Fecha y hora en formato internacional | ISODate          |

Tabla 36: Colección de datos para la entidad GAS\_TEST.

## 5. Implementación

Esta sección de la memoria del proyecto está dedicada a abordar el desarrollo del software que posteriormente se cargará tanto en las placas de desarrollo como en la Raspberry Pi.

El objetivo de nuestro proyecto era diseñar y construir un prototipo de sistema IoT funcional y completo capaz de monitorizar las deposiciones de la persona a monitorizar, para posteriormente enviarlas al sumidero de datos donde se procesarían e insertarían en la base de datos. Como se ha podido ver en las secciones anteriores, ya se dispone del hardware necesario para llevar a cabo nuestro objetivo, sin embargo, aún se necesita el software que se encargue de dotar al prototipo de las funcionalidades necesarias para tal propósito.

A continuación, se mostrará cómo podemos programar los microcontroladores escogidos de una manera sencilla. Se explicará el entorno de desarrollo que se ha utilizado para su programación, las ventajas respecto a otros entornos de desarrollo, y los pasos necesarios para configurar adecuadamente el entorno respecto a las placas de desarrollo escogidas.

### 5.1. Entorno de desarrollo

Como ya se ha comentado en secciones anteriores, se van a emplear dos módulos de desarrollo, uno basado en el microcontrolador ESP32 y el otro basado en el microcontrolador ESP8266. Actualmente, para desarrollar software destinado a estos microcontroladores se suele utilizar el IDE de Arduino. Este entorno de desarrollo es bastante sencillo y ofrece las funcionalidades básicas para el desarrollo de software, pero carece de funcionalidades más avanzadas. Es por esta razón que para este proyecto se ha utilizado Visual Studio Code.

Visual Studio Code es un editor de código, que mediante una serie de módulos o extensiones aumenta las funcionalidades que ofrece de serie. Para complementar las funcionalidades se han utilizado las extensiones de C++, Python, y PlatformIO IDE

[25], esta última es la que nos ofrece la capacidad de desarrollar software para los microcontroladores. Una vez se he descargado la extensión de PlatformIO [45], se va a proceder a explicar su uso.

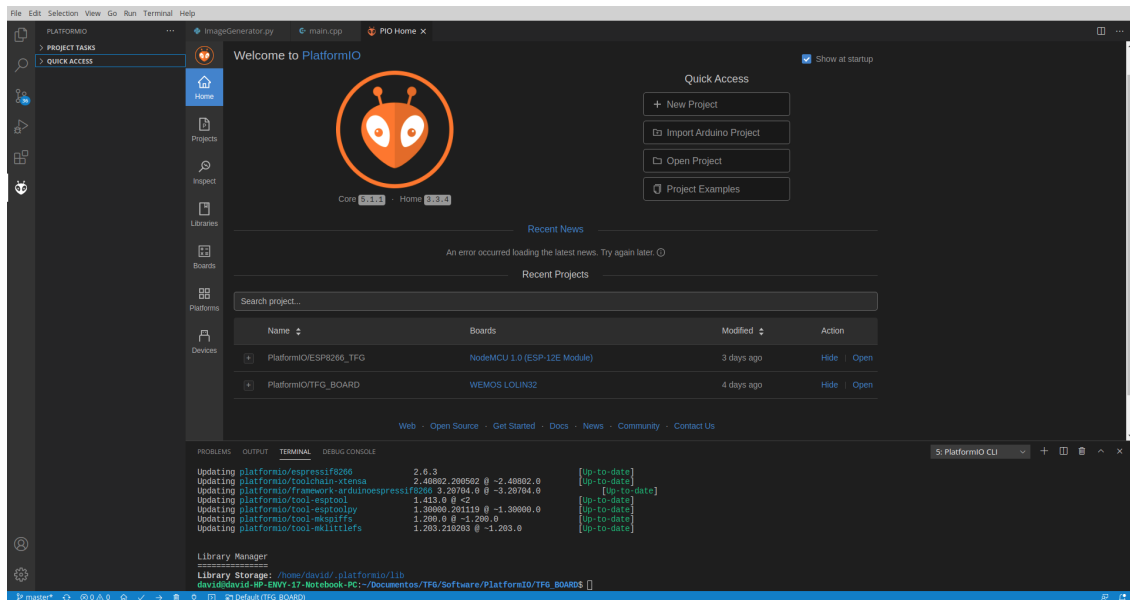


Figura 39: Ventana principal de PlatformIO

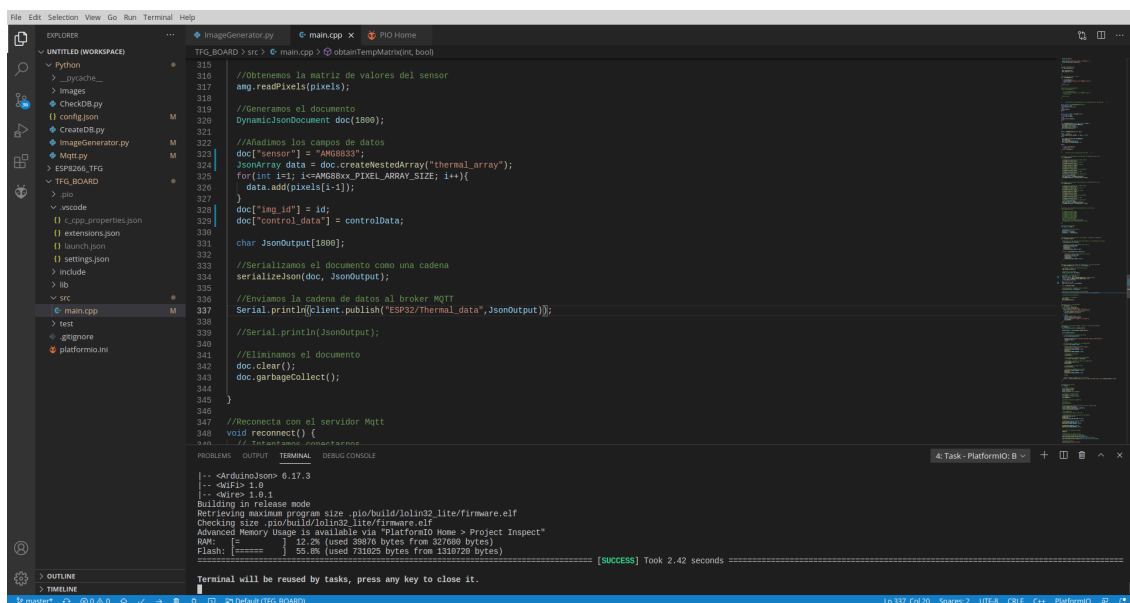


Figura 40: Visual Studio Code + PlatformIO IDE

En la ventana principal de la extensión dispondremos de accesos rápidos a la mayoría de funciones que nos ofrece.

- **Projects.** En esta sección se podrán crear nuevos proyectos o visualizar los ya existentes. Un punto clave de este IDE es que permite importar proyectos ya existentes de Arduino, simplificando la transición.
- **Libraries.** En esta sección se pueden añadir las diferentes bibliotecas necesarias para brindar funcionalidades adicionales o dar soporte a la gran variedad de sensores disponibles. Estas librerías se instalarán de manera global, es decir, estarán disponibles para todos los proyectos que se creen.
- **Boards.** Esta sección contiene todas las placas de desarrollo que actualmente están soportadas para programarlas, por tanto la primera tarea sería comprobar que disponemos de dicha compatibilidad. Como se verá en la siguiente sección, en el caso de que la placa escogida no se encuentre disponible entre las que se incluyen por defecto, se podrá añadir mediante la inclusión de su plataforma.
- **Platforms.** Desde esta sección podremos añadir más plataformas para aumentar el número de placas que soporta PlatformIO. Como nuestro proyecto utiliza dos placas de desarrollo basadas en el microcontrolador ESP32 Y ESP8266, será necesario añadir la plataforma Espressif 32 y Espressif 8266 respectivamente para poder realizar la programación de las mismas.
- **Devices.** Esta sección contiene los dispositivos que se encuentran conectados al equipo de desarrollo. Nos permitirá detectar si existe algún problema relacionado con la conexión con nuestras placas de desarrollo, como pueden ser la falta de drivers.

El esquema general que siguen todos los programas de Arduino está formado por dos partes. Estas partes tienen forma de función:

- **Setup:** Esta función se ejecuta automáticamente al iniciar nuestro programa. Su finalidad principal es inicializar variables y cargar los datos necesarios que necesitaremos a lo largo de nuestro programa.
- **Loop:** Esta función es la principal del programa. Se ejecuta a continuación de la función Setup, pero la principal diferencia reside en que se ejecuta de forma



cíclica. Aquí se ubicarán las diferentes acciones que tiene que realizar nuestro programa.

Adicionalmente a las funciones principales, se podrán declarar las funciones y variables que se requieran para dotar al programa de las funcionalidades deseadas.

## 5.2. Sumidero de datos

El sumidero de datos se corresponde con la parte del sistema encargado de recolectar toda la información proveniente de las placas de desarrollo. En este caso, dichas funciones se ubican en la Raspberry Pi.

### 5.2.1. MQTT

Como se ha comentado en secciones anteriores, la comunicación entre los dispositivos se realiza vía Wi-Fi, y utilizando el protocolo de comunicación MQTT [41]. MQTT es un protocolo de mensajería, desarrollado sobre TCP/IP, que se basa a su vez en el modelo de mensajería de publicación/suscripción. Los publicadores envían los mensajes a un determinado topic, los suscriptores reciben los mensajes de los topic a los que están suscritos y los agentes transmiten los mensajes de los publicadores a los suscriptores.

Los publicadores y suscriptores son clientes de MQTT, que solo se comunican con un agente de MQTT denominado broker.

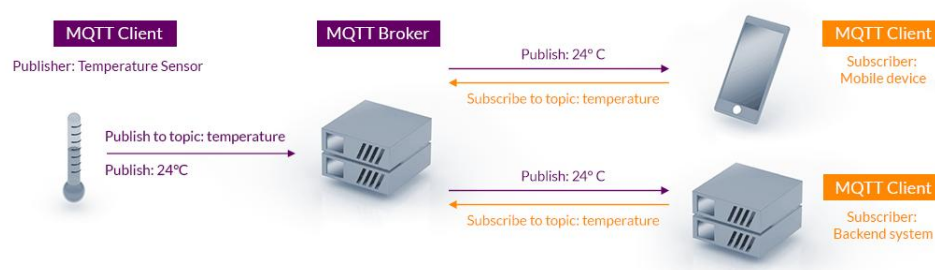


Figura 41: Modelo de comunicación MQTT [40]

Los clientes de MQTT pueden ser cualquier dispositivo o aplicación que se conecta a un agente de MQTT a través de una red. Los agentes de MQTT gestionan la recepción

de los mensajes de los publicadores y el envío de los mensajes a los suscriptores. El filtro que aplican los publicadores y suscriptores para determinar a quien le llegan los mensajes se denomina topic.

## ■ Topics

Los topics no son más que una cadena de texto en formato UTF-8 en la que se distingue entre minúsculas y mayúsculas. Estos pueden estar compuestos por uno o varios niveles, Figura 42, separados entre si mediante una barra inclinada.

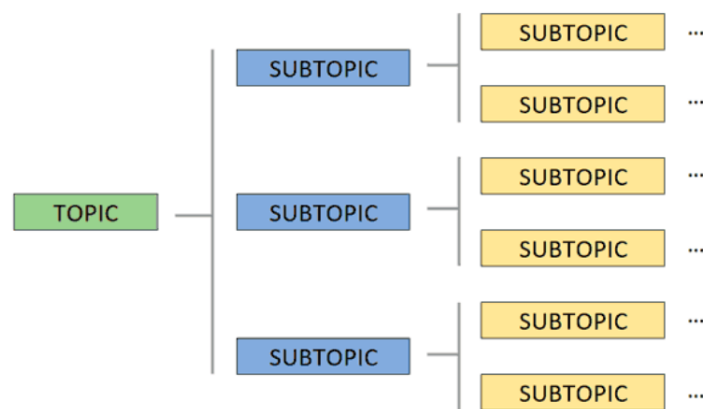


Figura 42: Niveles de los topics

Un ejemplo de topics serían los siguientes:

- Casa/Salon/Temperatura
- Casa/Salon/Humedad
- Casa/Sotano/Humedad
- Casa/Desvan/Temperatura

A la hora de suscribirnos a diferentes topics es posible utilizar Wildcards para suscribirse a múltiples topics. Existen dos tipos.

- Wildcard de nivel único. El carácter + puede ser utilizado para sustituir un único nivel.

- Wildcard de nivel múltiple. El carácter # puede ser utilizado para sustituir cualquier número de niveles, pero solo puede ser utilizado al final del topic.

## ■ Formato

En el protocolo MQTT se intercambian una serie de paquetes de control para llevar a cabo la comunicación. Estos paquetes pueden estar formados por tres partes en función del de su tipo.

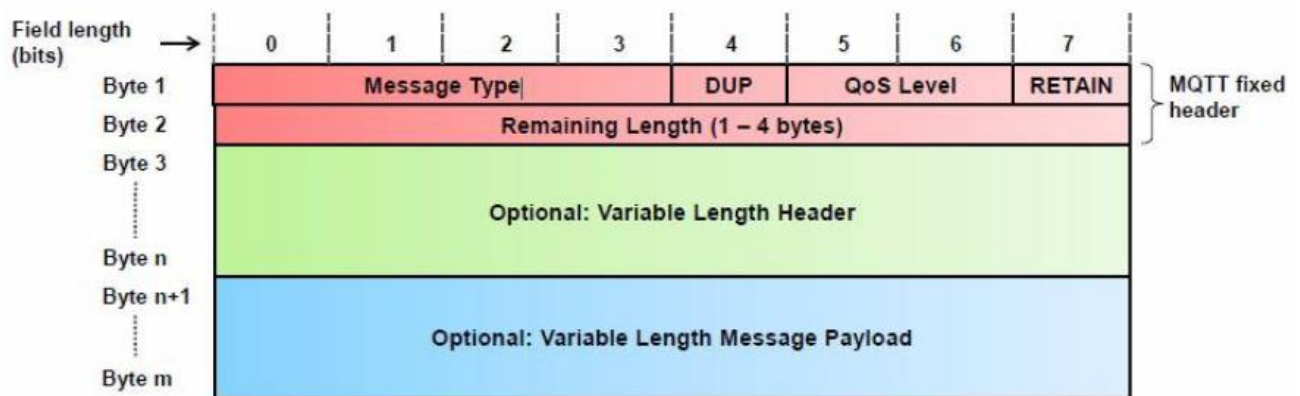


Figura 43: Estructura paquete MQTT [66].

La cabecera fija o fixed header es la parte inicial de los paquetes MQTT y es la única que se encuentra en todos los tipos. Entre la distinta información que aporta se encuentra el nivel QoS (Quality of Service), el tipo de mensaje o el número de bytes restantes del paquete incluyendo la cabecera variable y el payload.

La cabecera variable o variable header se encuentra presente en algunos de los paquetes MQTT. Dependiendo del tipo de mensaje o el QoS puede contener distintos campos que aporten información, como puede ser el nombre del topic.

La carga útil o payload es la última parte del paquete, es de longitud variable y contiene toda la información a enviar. Esta parte no se encuentra disponible en los paquetes enfocados al control de la conexión.

## ■ QoS

Otra de las características más relevantes de MQTT es la calidad del servicio o QoS. QoS nos permite gestionar la robustez en el envío de los mensajes mediante tres niveles posibles.

- **QoS 0** unacknowledged (at most one): El mensaje se envía una única vez. En caso de fallo es posible que el suscriptor no reciba el mensaje.
- **QoS 1** acknowledged (at least one): El mensaje se envía hasta que garantiza la entrega. En caso de fallo es posible que el suscriptor reciba mensajes duplicados.
- **QoS 2** assured (exactly one). Se garantiza que cada mensaje se entrega al suscriptor una única vez.

La elección de un nivel u otro de calidad del servicio dependerá de nuestro sistema. Un QoS mayor utilizará un mayor número de mensajes para realizar las comprobaciones, aumentando la carga del sistema. En nuestro caso particular, dada la naturaleza no imprescindible del contenido de los mensajes, hemos optado por utilizar un nivel de calidad del servicio QoS 0.

## ■ Seguridad

La seguridad es un tema muy importante a la hora de diseñar un sistema. El protocolo MQTT dispone de distintas medidas de seguridad que podemos adoptar para proteger las comunicaciones. Esto incluye soporte **SSL/TLS** (Secure Sockets Layer/Transport Layer Security), autenticación mediante usuario y contraseña o mediante certificados. El más recomendado es el uso de MQTT con TLS. Su implementación se realiza en el servidor y se debe usar el puerto TCP 8883, el cual permite la encriptación en el intercambio de mensajes usando los algoritmos de cifrado simétrico AES (Advanced Encryption Standard) o DES (Data Encryption Standard).

La implementación o no de estas medidas de seguridad están determinadas tanto por el tipo de proyecto, como por los dispositivos que se utilizarán. Las placas de desarrollo seleccionadas aún siendo notablemente potentes en comparación con otros modelos, pueden llegar a presentar problemas al tener que lidiar con la gestión de la seguridad y el alto flujo de mensajes. Además, todos los dispositivos se ubican en una red local, dificultando por tanto el acceso de terceros al flujo de datos entre los diferentes elementos. Es por este conjunto de razones, que se ha descartado el uso de estas técnicas para este proyecto.

Una vez hemos visto las principales características del protocolo MQTT, vamos a explicar como se ha gestionado su uso en el sumidero de datos.

El primer paso es seleccionar un broker MQTT. Un broker MQTT es como se le denomina al agente que se encarga de gestionar el flujo de mensajes entre los clientes. En la actualidad existen multitud de agentes open-source que podemos utilizar. Entre ellos se encuentran Eclipse Mosquitto, HBMQTT, EMQ [20] o ActiveMQ [10].

- **Mosquitto:** Eclipse Mosquitto es un agente de mensajes open source que implementa las versiones 5.0, 3.1.1 y 3.1 del protocolo MQTT. Mosquitto es liviano y adecuado para su uso en todos los dispositivos.
- **MBMQTT:** Este broker MQTT soporta la especificación 3.1.1 al completo. Tiene autenticación basada en archivos de contraseñas y puede utilizar SSL (Secure Sockets Layer) sobre TCP (Transmission Control Protocol) o websockets.
- **EMQ:** Erlang MQTT Broker es un intermediario de mensajes MQTT distribuido, masivamente escalable y altamente extensible escrito en Erlang/OTP. Soporta el uso de MQTT es sus versiones 3.1.1 y 3.1, además, soporta otros protocolos como pueden ser STOMP (SimpleText Orientated Messaging Protocol) [53] , MQTT-SN (MQTT for Sensor Networks) [27] o CoAP (Constrained Application Protocol) [16].

- **ActiveMQ:** Apache ActiveMQ es un servidor de mensajería open source. Soporta AMQP (Advanced Message Queuing Protocol) [42] , STOMP y MQTT 3.1. Incorpora además una API RESTful, que permite a cualquier dispositivo compatible con web publicar o consumir mensajes utilizando HTTP POST o GET.

Para este proyecto se ha seleccionado Eclipse Mosquitto, que es un broker suficientemente ligero para integrarlo en una Raspberry Pi. Una vez se ha seleccionado el broker que queremos utilizar, debemos definir cuales serán los topics necesarios para la comunicación y los clientes.

Para realizar nuestro proyecto son necesarios 3 clientes, ESP32, ESP8266 y Raspberry Pi. Como veremos más adelante todos estos clientes ejercen de publicadores y de suscriptores.

Una vez hemos determinado los clientes, procedemos a definir los diferentes topics para organizar los mensajes.

- **ESP32/Thermal\_data.** Este topic es el que contiene la información correspondiente a los valores de temperatura del sensor ubicado en la placa de desarrollo ESP32. La placa ESP32 ejerce de publicador y la Raspberry Pi de suscriptor.
- **ESP8266/Gas\_data.** Este topic contiene los valores obtenidos por el sensor de gas. La placa ESP8266 ejerce de publicador y la Raspberry Pi de suscriptor.
- **ESP32/Use\_time.** En este topic la placa ESP32 publica cuando se abre o cierra la tapa del inodoro. La placa ESP32 ejerce de publicador y la Raspberry Pi de suscriptor.
- **ESP8266/Gas\_test.** Este topic se utilizó para publicar la información de la prueba de gases con el sensor MQ135 y MQ2. La placa ESP8266 ejerce de publicador y la Raspberry Pi de suscriptor.

- **Pi/Config/ESP32.** Contiene la información de configuración de la placa ESP32. La Raspberry Pi ejerce de publicador y la ESP32 de suscriptor.
- **Pi/Config/ESP8266.** Contiene la información de configuración de la placa ESP8266. La Raspberry Pi ejerce de publicador y la ESP8266 de suscriptor.

### 5.2.2. Transformación de los datos de temperatura en imágenes

Como hemos explicado con anterioridad, el conjunto de datos que nos devuelve la cámara térmica se corresponde con una matriz 8x8 que contiene los valores de temperatura obtenidos. Realizar un análisis visual de los datos en este tipo de formato puede resultar tedioso, por esta razón se ha implementado un sistema que transforma dichas matrices en imágenes.

El primer paso que realiza el sistema consiste en cargar los parámetros necesarios para su funcionamiento. Estos parámetros están definidos en el archivo de configuración config-img.json, y son los siguientes:

- **HIGHER\_COLOR.** Este parámetro se encarga de determinar el color del rango superior. Viene expresado en hexadecimal. El valor por defecto es #FF0000, que se corresponde con el color rojo.
- **LOWER\_COLOR.** Este parámetro se encarga de determinar el color del rango inferior. Viene expresado en hexadecimal. El valor por defecto es #4B0082, que se corresponde con el color morado.
- **HIGHER\_TEMP.** Se corresponde con la máxima temperatura que puede representar nuestra imagen.
- **LOWER\_TEMP.** Se corresponde con la mínima temperatura que puede representar nuestra imagen.
- **COLOR\_DEPTH.** Este valor indica el número de posibles colores que existen entre el límite superior y el inferior.
- **IMAGE\_RESOLUTION.** Define el valor de la resolución de la imagen final.

Una vez que hemos visto cuales son los principales parámetros de nuestro sistema, podemos empezar a explicar su funcionamiento.

- El primer paso consiste en extraer el rango de colores que hay contenido entre los colores que habíamos definido previamente. Se extraerán exactamente el número de colores definidos en el parámetro COLOR\_DEPTH.

---

**Algorithm 1** generateImage(datArray)

---

$lower \leftarrow obtainColor(LOWER\_COLOR)$

$colorsArray \leftarrow list(lower.rangeTo(HIGHER\_COLOR, COLOR\_DEPTH))$

---

- El siguiente paso consiste en la generación de una imagen vacía y la transformación de los valores de temperatura de la matriz, aplicando una normalización de los mismos según el rango que hayamos definido.

---

**Algorithm 2** generateImage(datArray)

---

$blank\_image \leftarrow image$

$pixelsArray \leftarrow datArray$

**for** every pixel in pixelsArray **do**

$newPixel \leftarrow (pixel - LOWER\_TEMP) * (COLOR\_DEPTH - 0) / (HIGHER\_TEMP - LOWER\_TEMP) + 0$

$pixelsArray \leftarrow add(newPixel)$

**end for**

$modifiedPixels \leftarrow BlankArray$

**for** every pixel in pixelsArray **do**

$colour \leftarrow obtainColor(min(max(0, pixel), COLOR\_DEPTH))$

$modifiedPixels \leftarrow add(colour)$

**end for**

---

- En la siguiente fase, mediante funciones de las librerías utilizadas se realiza la asignación del color en función de su valor. Estos colores son asignados a la imagen vacía que creamos con anterioridad.



**Algorithm 3** generatelmage(datArray)

---

```

for every pixel in modifiedPixel do
    blank_image  $\leftarrow$  add(observeOnRGBColor(pixel))
end for

```

---

- Una vez llegados a este punto, ya se tiene una imagen de 8x8 píxeles. Esta resolución propicia que la imagen obtenida sea demasiado pequeña para visualizarla correctamente. Es por ello, que se le aplica un proceso de reescalado a la imagen para conseguir la resolución que previamente habíamos definido en los parámetros de configuración. Para el proceso reescalado se ha aplicado interpolación bicúbica. Este tipo de técnica toma como muestra 16 píxeles para determinar el color del nuevo pixel.

**Algorithm 4** generatelmage(datArray)

---

```

image  $\leftarrow$  resize(Bicubic, IMAGE_RESOLUTION, blank_image)

```

---

En la Figura 44 podemos observar el array de datos de temperatura del que partimos, y la imagen resultante una vez aplicamos las diferentes funciones.

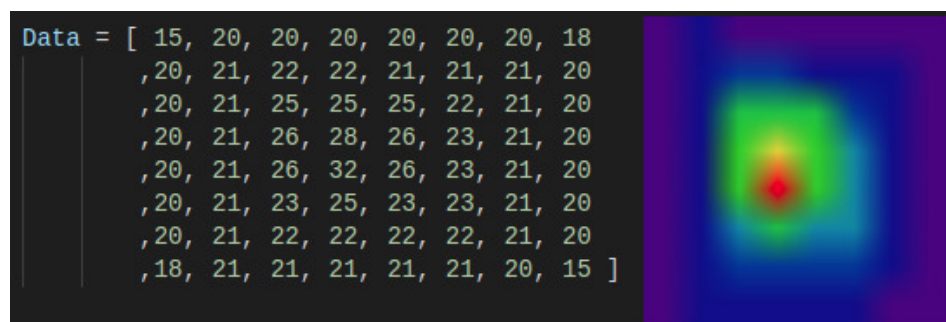


Figura 44: Datos de entrada e imagen obtenida.

### 5.3. Sistema para la recogida de datos

Como se ha comentado en la sección anterior, para desarrollar el código que posteriormente grabaremos en las placas de desarrollo utilizamos el entorno de desarrollo de PlatformIO configurado específicamente para la placa. Para configurar la placa es necesario editar un archivo de configuración cuyo nombre es **platformio.ini**. En este caso, los parámetros que se han declarado son los siguientes:

- platform = espressif32. Este parámetro le indica al IDE con que plataforma de las existentes estamos trabajando.
- board = lolin32. Indica el modelo específico de placa de desarrollo que vamos a programar.
- framework = arduino. Nos permite indicar con que framework queremos trabajar. Para poder utilizar las librerías existentes para Arduino es necesario indicar que queremos utilizar el framework de arduino.
- monitor\_speed = 115200.
- board\_build.f\_cpu = 160000000L. Este parámetro determina la frecuencia de la CPU, en nuestro caso la hemos fijado a 160 MHz para reducir el consumo de energía.
- lib\_deps = knolleary/PubSubClient @ 2.8, adafruit/Adafruit AMG88xx Library @ 1.1.0, bblanchon/ArduinoJson @ 6.17.3. En este parámetro indicamos las librerías necesarias para utilizar los distintos sensores y funcionalidades que requiramos.

Una vez realiza la configuración ya podemos empezar a desarrollar el código necesario para nuestro dispositivo.

### 5.3.1. Sistema maestro-esclavo

Para implementar la sincronización entre las dos placas de desarrollo hemos utilizado un esquema maestro-esclavo. La placa ESP32 actúa como maestro y la placa ESP8266 como esclavo. El esquema de funcionamiento es realmente simple.

- Cuando la persona a monitorizar abre la tapa del inodoro, la placa ESP32 envía una señal a la ESP8266 para sacarla del modo Deep Sleep. Una vez se despierta la placa ESP8266, empezará a recoger los datos.
- Cuando la persona a monitorizar baja la tapa, la placa ESP32 volverá a enviar una señal para que la ESP8266 finalice la recolección de datos y vuelva a entrar en modo Deep Sleep.

Para implementar el sistema hemos utilizado cuatro pines:

- GPIO16, ESP32. Desde este pin se envía la señal para despertar a la placa.
- GPIO17, ESP32. Enviamos la señal de que finalice la recolección de datos.
- RST, ESP8266. En este pin recibimos la señal que despierta a la placa.
- D5, ESP8266. Utilizado para recibir la señal que finaliza la recogida de datos.

El esquema de conexiones es el siguiente:

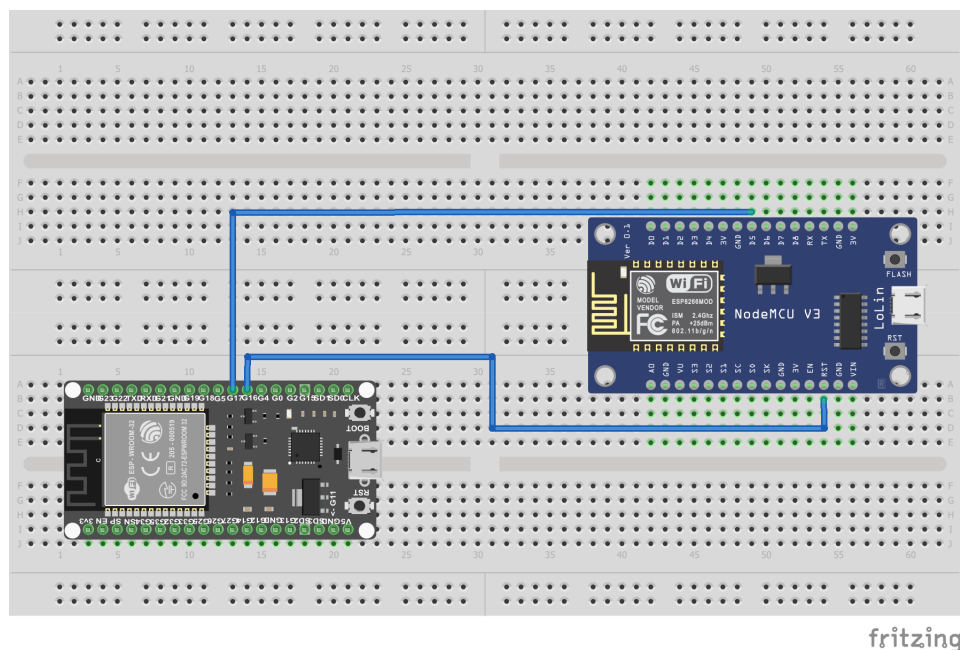


Figura 45: Esquema de conexiones para el sistema maestro-esclavo.

### 5.3.2. Transmisión de datos

Una de las necesidades de nuestro proyecto consiste en transmitir la información recolectada por los distintos sensores a nuestra Raspberry Pi. Para ello deberemos implementar una serie de funciones que nos permitirán conectar nuestros dispositivos a una red Wi-Fi. Para esta labor existen bibliotecas especializadas que nos permiten realizar la conexión de una manera sencilla, en nuestro caso importaremos el archivo de cabecera *WiFi.h* correspondiente a la librería WiFi. Para configurar la conexión tendremos que especificar dos parámetros básicos:

- **SSID**: identificador de la red Wi-Fi con la que queremos establecer conexión.

- **Password:** contraseña de la red con el SSID especificado.

Estos parámetros son los esenciales, pero podrían definirse más, tales como la dirección ip, la puerta de enlace, etc. En este caso se ha declarado un parámetro adicional:

- **Time out:** el tiempo máximo para realizar la conexión. En caso de sobrepasarlo abortamos el intento de conexión.

Con estos parámetros ya podemos proceder a configurar la conexión a la red Wi-Fi.

Realizar la configuración a través de la librería resulta muy sencillo, únicamente deberemos proporcionarle el SSID de la red y la contraseña. Durante la configuración también le hemos indicado el modo de funcionamiento en modo estación (STA, Station), es decir, como un dispositivo que se conecta a un punto de acceso (AP, Access Point). También es posible configurar la placa en modo AP o punto de acceso, lo que nos permite crear una red Wi-Fi con nuestra placa de desarrollo.

### 5.3.3. MQTT

Como ya hemos comentado anteriormente, MQTT (Message Queuing Telemetry Transport) es un protocolo red basado en la publicación y suscripción. Es un protocolo muy ligero, ideal para dispositivos con bajos recursos como nuestras placas de desarrollo.

Para implementar el protocolo en nuestra aplicación vamos a utilizar una librería [5] especialmente desarrollada para tal fin. Esta librería nos proporciona los métodos necesarios para poder utilizar el protocolo, sin embargo carece de algunas funciones avanzadas, como puede ser el control de acceso al servidor mediante certificados o el QoS de los mensajes. Si estas funcionalidades son requeridas por parte de nuestra aplicación será necesario utilizar otras librerías que las soporten o bien realizar una implementación de las mismas.

Nuestro primer paso para usar la librería será cargar el archivo de cabecera correspondiente, en este caso "PubSubClient.h". Una vez declarado deberemos crear un objeto de la librería PubSubClient, y pasarle como parámetro un objeto del tipo WiFiClient. Este objeto es necesario ya que la librería delega el envío y la recepción de los mensajes en la librería correspondiente a la conexión Wi-Fi.

Los parámetros que deberemos configurar son los siguientes:

- **MQTT\_SERVER.** Este parámetro le indicará a la librería en que dirección IP de nuestra red se aloja el servidor o broker correspondiente al protocolo MQTT.
- **MQTT\_PORT.** Se corresponde con el puerto que nuestro servidor utiliza para el protocolo. Por defecto es el 1883 para conexiones no cifradas.

Estos parámetros que hemos configurado son los indispensables para utilizar la librería, pero existen una gran variedad de ellos que nos permitirán configurar aspectos más avanzados. Uno de estos parámetros es el buffer size, que nos permite establecer el tamaño máximo de los mensajes. En nuestro caso los hemos establecido a 512 bytes, lo que nos permitirá trabajar con mensajes de mayor tamaño.

Las funciones que deberemos implementar para poder realizar un uso de la librería son las siguientes.

- **Callback.** Esta función es la encargada de gestionar los mensajes recibidos de los diferentes topics en los que hemos realizado la suscripción, además, dentro de esta función podremos realizar el tratamiento de los mensajes. Sus parámetros de entrada se corresponden con el tema o topic del mensaje, su carga útil o mensaje, y la longitud del mismo. Para poder verificar si tenemos mensajes entrantes deberemos llamar regularmente a la función *client.loop()*, que se encargará de llamar a callback en caso de que haya mensajes pendientes.
- **Reconnect.** Esta función nos permite verificar el estado de nuestra conexión. En caso de que no se encuentre conectada intentará conectarse con el identificador proporcionado, en nuestro caso *ESP32Client*. Este identificador debe ser único

en nuestra red, puesto que si existieran varios dispositivos conectados al mismo tiempo con el mismo identificador el sistema expulsaría al de mayor antigüedad. También realizaremos la suscripción a los diferentes topics en esta función, con el objetivo de renovar la suscripción de manera automática en caso de una pérdida de conexión.

- **Publish.** Sus parámetros de entrada se corresponden con el tema o topic del mensaje que queremos enviar y el contenido del mismo. Nos proporciona además un resultado de salida que nos indica mediante un booleano si el mensaje se ha podido enviar satisfactoriamente.

#### 5.3.4. Serialización de los datos

El dispositivo debe transmitir los datos recopilados a una plataforma externa mediante una conexión Wi-Fi. Serializar los objetos con los que se trabaja en memoria permite convertir estructuras complejas en cadenas de texto, permitiendo enviar todos los datos en un único mensaje, reduciendo considerablemente el tratamiento posterior de los mismos y reduciendo la carga en el servidor MQTT. Para el formato de texto de los datos se ha empleado JSON (JavaScript Object Notation). Utilizar JSON conlleva una serie de ventajas:

- Es autodescriptivo y fácil de entender.
- Es más ligero (bytes) en las transmisiones.
- Se parsea más rápido.
- Velocidad de procesamiento alta.

En nuestro caso hemos optado por utilizar la librería **ArduinoJson** [12] para la serialización de los datos.

El primer paso para utilizar la librería es declararla en el archivo de configuración y cargarla en nuestro sketch. Esto último lo haremos incluyendo el archivo de cabecera *ArduinoJson.h* en nuestro sketch. Una vez hemos incluido el archivo de cabecera será necesario crear un objeto de la librería para poder serializar los datos.

La librería *ArduinoJson* nos permite determinar si queremos guardar los objetos en el stack o en el heap mediante los objetos *StaticJsonDocument* y *DynamicJsonDocument* respectivamente. Será necesario indicar en ambos casos la cantidad de memoria RAM necesaria para alojar el documento. Este paso es de vital importancia puesto que un valor insuficiente provocará que nuestros datos no se registren correctamente, pero un valor demasiado alto resultará en un desperdicio de memoria que podría ocasionar fallos en la ejecución. Para explicar su funcionamiento se van a serializar los valores obtenidos por el sensor de gas.

1. Declaramos el objeto, asignándole el espacio previamente calculado.

---

```
DynamicJsonDocument doc(80);
```

---

2. Añadimos el modelo de sensor.

---

```
doc["Sensor"] = "MQ2";
```

---

3. Añadimos el valor en raw obtenido por el sensor.

---

```
doc["Raw_Data"] = 202;
```

---

4. Añadimos el valor en ppm.

---

```
doc["PPM"] = 438;
```

---

5. Generamos el JSON que vamos a enviar.

---

```
serializeJson(doc, JsonOutput);
```

---

6. El resultado obtenido sería el siguiente.

```
{"Sensor":"MQ2",Raw_Data":202,"PPM":438}
```

Una vez tenemos la cadena de texto en el formato correspondiente ya podemos proceder a enviarla por MQTT mediante las funciones que hemos explicado anteriormente.

Para realizar el proceso inverso, es decir dada una cadena de texto convertirla a objetos, los pasos a seguir son ligeramente distintos como veremos a continuación:

1. Definimos una cadena de texto con el siguiente formato.

---

```
json = "{\"WAKEUP_TIME\": \"60\", \"TIME\": 20, \"THERMAL_SAMPLES\": 20}";
```

---

2. Creamos el documento que contendrá los datos.

---

```
StaticJsonDocument<250> doc;
```

---

3. Deserializamos la cadena de texto, comprobando si se ha producido algún error.

4. En caso de que no se hayan producido errores ya podemos acceder al contenido.

---

```
int new_wake_time = doc["WAKEUP_TIME"];  
int new_time = doc["TIME"];  
int new_samples = doc["THERMAL_SAMPLES"];
```

---

Como hemos podido ver, el uso de esta librería facilita el manejo de la información.

### 5.3.5. Manejo de la memoria no volátil

Anteriormente hemos visto como podíamos recuperar información mediante la deserialización de las cadenas de textos recibidas mediante MQTT. Este proceso nos permitía obtener los parámetros de configuración necesarios para el funcionamiento de nuestro dispositivo, pero dichos parámetros se almacenan en la memoria RAM, y son eliminados cada vez que la placa se reinicia. Para poder utilizar dichos parámetros después de un reinicio deberemos utilizar la memoria no volátil de la que disponen las placas, es decir, la EEPROM (Electrically Erasable Programmable Read-Only Memory).

Para trabajar con la memoria EEPROM disponible en nuestra placa de desarrollo existen dos alternativas en forma de librerías, *Preferences* y *EEPROM*. Aunque las dos librerías nos permiten escribir y leer información de nuestra memoria EEPROM,



*Preferences* dispone de características más avanzadas. Entre dichas características se encuentran las siguientes:

- **Wear leveling.** Como sabemos, las memorias flash disponen de una vida útil determinada por un número máximo de escrituras que se pueden realizar antes de que la memoria se degrade. Wear leveling se encarga de repartir homogéneamente la información entre las distintas celdas que conforman la memoria para igualar el nivel de desgaste.
- **Abstracción.** Mediante el uso de esta librería podemos realizar escrituras y lecturas de los datos sin tener que preocuparnos de gestionar la memoria.

Para poder utilizar las funciones de la librería será necesario declarar en nuestro sketch el archivo de cabecera *Preferences.h*. Una vez lo hemos declarado ya podemos empezar a utilizarlo. Crearemos un objeto del tipo *Preferences*, con el cual podremos acceder a las distintas funciones que ofrece.

### 5.3.6. Implementación del sensor amg8833

Para poder controlar el sensor amg8833 se van a utilizar las librerías especialmente diseñadas para este propósito. En este caso las librerías que se van a utilizar son las desarrolladas por el fabricante del sensor, Adafruit. Para poder utilizarlas deberemos declarar en nuestro sketch el archivo de cabecera *Adafruit\_AMG88xx.h*.

Una vez declarada la librería se deberán emplear cuatro pines, dos para proporcionar alimentación al sensor y dos para poder realizar la lectura de los datos:

1. **PIN\_I2C\_SDA:** es el pin conectado a la patilla SDA del sensor. En nuestro caso hemos utilizado la patilla 18 de nuestra placa de desarrollo.
2. **PIN\_I2C\_SCL:** es el pin conectado a la patilla SCL del sensor. Para ello hemos utilizado la patilla 22 de nuestra placa de desarrollo.

El esquema de conexiones es el siguiente:



- Obtenemos el array de valores de temperatura mediante los métodos que nos proporciona la librería.
- Creamos un documento Json.
- Añadimos al documento los campos de identificación y el array de valores.
- Serializamos el documento como una cadena de texto y procedemos a enviarla mediante MQTT.

**Algorithm 5** obtainTempMatrix(id, controlData)

---

```

pixels ← readPixels(pixels)
maxTemp ← pixels[0]
minTemp ← pixels[0]
JsonDocument ← ∅
JsonDocument["sensor"] ← "AMG8833"
JsonArray ← newArray("thermal_array")
for every pixel in pixels array do
    JsonArray ← pixel
    if pixel greater than maxTemp then
        maxTemp ← pixel
    else if pixel lower than minTemp then
        minTemp ← pixel
    end if
end for
JsonDocument["img_id"] ← id
JsonDocument["control_data"] ← controlData
JsonOutput ← serialize(JsonDocument)
publishMQTT("ESP32/Thermal_data", JsonOutput)

```

---

Como hemos podido ver, el proceso para obtener los valores del sensor es realmente sencillo a través de la biblioteca, únicamente deberemos llamar a la función *readPixels(pixels)*, proporcionándole el array que habíamos definido previamente.

### 5.3.7. Implementación del sensor MMA8451

Como se ha comentado en capítulos anteriores, este sensor nos permite medir la aceleración, es decir se trata de un acelerómetro. Este sensor ejerce un papel clave en el funcionamiento del dispositivo, debido a que es el responsable de iniciar la fase de recogida de datos en base a las acciones de las personas a monitorizar.

Durante las primeras fases del desarrollo se optó por utilizar una librería [8], parcialmente compatible con nuestro sensor. Entre las principales modificaciones que

se realizaron se encuentran la supresión de la verificación del componente, y la asignación de la dirección de memoria de nuestro sensor. Este tipo de modificaciones no son recomendables, debido a que al alterar localmente una librería, estos cambios podrían revertirse en una actualización posterior de la misma. Además de estas modificaciones, la librería carece de ciertas funciones más avanzadas. Dichas funciones se corresponden con la capacidad de utilizar el sensor mediante interrupciones, es decir, configurar el sensor para que superado cierto umbral de aceleración lance una señal automáticamente a la placa, sin la necesidad de estar comprobando de manera reiterada la aceleración que está experimentando el sensor. Es por esta razón, que para controlar el sensor se ha optado por configurar el sensor a nivel de registros. Esto ha sido posible debido a la existencia de un datasheet (manual) del sensor [62], que detalla el funcionamiento interno del sensor, siendo posible editar su comportamiento escribiendo bytes en las direcciones de memoria indicadas.

Para poder configurar el sensor será necesario conectarlos a los pines de alimentación, al pin designado para la interrupción (GPIO4) y a los pines I2C de nuestra placa de desarrollo (GPIO18 y GPIO22).

El esquema de conexiones es el siguiente:

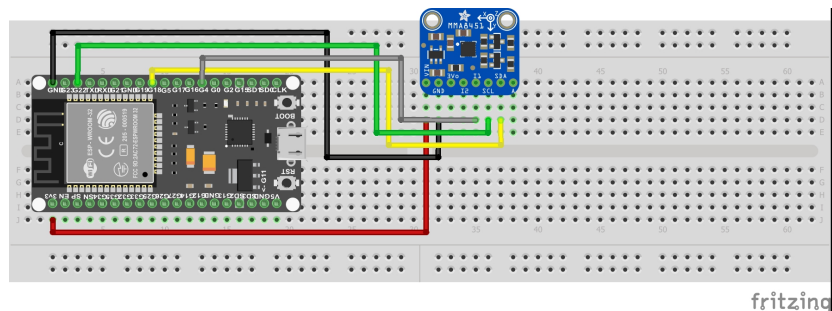


Figura 47: Esquema de conexiones para el sensor mma8451.

Una vez esté conectado, será necesario obtener la dirección del sensor, esto lo podemos realizar mediante un programa denominado I2C scanner. También será necesario implementar una serie de funciones para poder escribir y leer valores de los registros del sensor. Dichas funciones han sido extraídas directamente de la librería

mencionada anteriormente, por tanto la auditoría de las funciones les pertenece a los creadores de la misma.

---

```
void writeRegister8(uint8_t reg, uint8_t value) {
    Wire.beginTransaction(ADDR_MMA8451);
    i2cwrite((uint8_t)reg);
    i2cwrite((uint8_t)(value));
    Wire.endTransmission();
}

uint8_t readRegister8(uint8_t reg) {

    #ifdef __SAM3X8E__
        Wire.requestFrom(_i2caddr, 1, reg, 1, true);
    #else
        Wire.beginTransaction(ADDR_MMA8451);
        i2cwrite(reg);
        Wire.endTransmission(false);
        Wire.requestFrom(ADDR_MMA8451, 1);

    #endif

    if (!Wire.available())
        return -1;
    return (i2cread());
}
```

---

Como podemos ver, la función writeRegister8 toma por parámetro el registro donde queremos realizar la escritura, y el valor que queremos escribir. Es necesario indicar mediante la librería Wire que queremos iniciar una transmisión de datos a la dirección donde se ubica el sensor.

Una vez ya disponemos de las funciones necesarias para escribir y leer los registros, ya podemos proceder a realizar la configuración del sensor:

---

```
void interruptMotionX(){  
  
    //Configura el dispositivo en modo reposo  
    writeRegister8(0x2A,0x18);  
  
    //Configura la deteccion en el eje X  
    writeRegister8(0x15,0xC8);  
  
    //Fija el umbral >0,7g  
    writeRegister8(0x17,0x0B);  
  
    //Elimina falsas lecturas  
    writeRegister8(0x18,0x0A);  
  
    //Habilita las interrupciones por movimiento  
    writeRegister8(0x2D,0x04);  
  
    //Liga la interrupcion al pin INT1  
    writeRegister8(0x2E,0x04);  
  
    //Configura el dispositivo en modo activo  
    uint8_t CTRL_REG1_Data;  
    CTRL_REG1_Data = readRegister8(0x2A);  
    writeRegister8(0x2A, CTRL_REG1_Data | 0x01);  
  
}
```

---

Una vez llamamos a la anterior función, el sensor quedaría listo para enviar una señal de interrupción cuando detecte que se superan los 0,7g en el eje X. La configuración en el eje Z sería equivalente, únicamente deberíamos sustituir el byte correspondiente a la dirección de memoria 0x15.

### 5.3.8. Implementación del buzzer

Para poder utilizar el buzzer necesitaremos 3 pines:

- **GND y 3v3.** Servirán para proporcionarle alimentación al sensor.
- **GPIO26.** Este será el pin al que conectaremos la patilla marcada con la letra S en nuestro buzzer.

El esquema de conexiones es el siguiente:

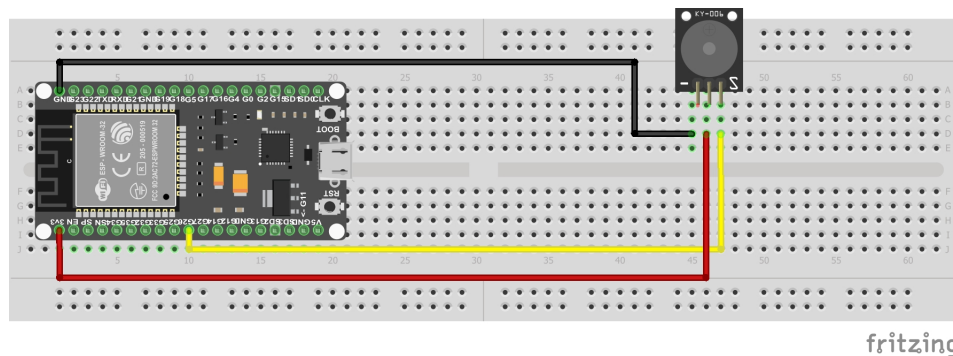


Figura 48: Esquema de conexiones para el buzzer.

### 5.3.9. Interrupciones y Deep Sleep

Las interrupciones son una característica fundamental de este tipo de placas, las cuales nos permiten interrumpir el ciclo habitual de nuestro programa para realizar acciones cortas. Esto lo podemos aprovechar también para sacar a nuestra placa de un modo especial denominado Deep Sleep [51], que veremos más adelante.

El primer paso que deberemos realizar será declarar el pin en el que queremos capturar la excepción, en nuestro caso el GPIO4. Es necesario tener en cuenta que no todos los pines digitales soportan el uso de interrupciones. A continuación, configuramos el pin como entrada, esto lo podemos realizar mediante la función `pinMode()`, indicándole el pin correspondiente y el modo de funcionamiento, que puede ser `INPUT` o `OUTPUT`, entrada o salida respectivamente. En este caso, aprovechamos también para llamar a la función que se encargará de configurar la interrupción como hemos visto anteriormente. El siguiente paso se corresponde con la captura de la interrupción.

Para capturar la interrupción será necesario utilizar la función `attachInterrupt()`, la cual requiere de tres parámetros, el pin en que se producirá la interrupción, la función

que ejecutará, y el estado que genera la interrupción. En nuestro caso la interrupción se genera cuando se produce una bajada de voltaje en el pin correspondiente.

La función ISR (Interruption Service Rutine) será ejecutada en el momento en el que se produzca la interrupción. Esta función debe ser lo más rápida posible, ya que altera el flujo principal del programa. Es por esta razón que se desaconseja el uso de delays dentro de ella y, además, la declaración de la función va precedida de *IRAM\_ATTR*, esto indica que la función se cargará en la memoria RAM en lugar de la memoria flash, lo que reducirá la latencia.

Una vez hemos realizado esta serie de acciones, nuestro programa ya puede realizar la captura y el tratamiento de las interrupciones que se produzcan en el pin indicado.

### Modos de funcionamiento

El chip ESP32 dispone de varios módulos en su interior. Dichos módulos pueden ser desactivados en función del modo de funcionamiento en el que se encuentre el propio chip.



Figura 49: Módulos del chip ESP32 [31].

A continuación vamos a ver los distintos modos de funcionamiento que ofrece el ESP32 [31] .



- **Active Mode.** En este modo de funcionamiento, todas las funciones de nuestro ESP32 están disponibles. Normalmente el consumo energético se sitúa en los 160 - 240 mA. Si utilizamos activamente las conexiones Wi-Fi y Bluetooth, podemos llegar a tener picos de consumo de aproximadamente 790 mA. Si no configuramos nuestra placa, este modo es el que se aplica por defecto.
- **Modem Sleep.** Este modo de funcionamiento deshabilita todas las funciones relacionadas con la conexión inalámbrica. Su consumo energético oscila entre los 3 -20 mA.
- **Light Sleep.** En este modo, los núcleos y la memoria del ESP32 se encuentran en un estado de reposo. La placa puede disminuir su consumo hasta los 0,8 mA.
- **Deep Sleep.** En este modo de funcionamiento, los núcleos y la memoria principal del ESP32 quedan deshabilitados, únicamente queda operativa una pequeña parte de la memoria que se conoce como RTC (Real Time Clock) memory, lo que nos permitirá preservar algunos datos. El co-procesador de bajo consumo así como el módulo RTC permanecerán activos. El consumo se reduce hasta los 10  $\mu$ A.
- **Hibernation.** Este modo deshabilita el co-procesador de bajo consumo y gran parte de las partes del módulo RTC. El consumo se reduce hasta los 2,5  $\mu$ A.

Como hemos podido ver, nuestra placa de desarrollo dispone de varios modos de funcionamiento que pueden resultar muy útiles en función de nuestro propósito. Nosotros hemos escogido Deep Sleep como modo de funcionamiento primario. Nos hemos decantado por esta opción en lugar de la hibernación debido al hecho de que al deshabilitar la mayoría de funciones del módulo RTC, ya no es posible despertar a nuestra placa mediante interrupciones externas, es decir, la única forma de despertar a la placa de la hibernación sería mediante un temporizador interno.

Una vez hemos visto los modos que nos ofrece la placa ya podemos empezar a utilizar el modo de Deep Sleep.

Nuestro primer paso consistirá en definir las interrupciones:

- **esp\_sleep\_enable\_ext0\_wakeup.** Esta función nos permite despertar a nuestra placa cuando se detecta un estado de tensión alto o bajo en un pin determinado.
- **esp\_sleep\_enable\_timer\_wakeup.** Esta función nos permite despertar a nuestra placa después de un tiempo determinado. Para poder usarla deberemos pasarle como parámetro el tiempo expresado en microsegundos.

Para sacar a nuestra placa del modo Deep Sleep utilizamos el pin de interrupción de nuestro acelerómetro y un estado de tensión bajo. También hemos realizado la configuración para que pasados 60 segundos la placa se reactive automáticamente. Nuestro siguiente paso consistirá en poner nuestra placa en el modo Deep Sleep. Esto lo podremos realizar mediante la función *esp\_deep\_sleep\_start*.

Una vez se ejecute esta función la placa permanecerá a la espera de que se produzca alguna de las dos condiciones definidas previamente.

Para determinar la causa deberemos utilizar la función *esp\_sleep\_get\_wakeup\_cause* y mediante un switch realizar el tratamiento adecuado. En nuestro caso hemos determinado tres tratamientos diferentes:

- **ESP\_SLEEP\_WAKEUP\_EXT0.** Se produce cuando el acelerómetro alcanza la aceleración indicada en el eje previamente definido.
- **ESP\_SLEEP\_WAKEUP\_TIMER.** Cuando se alcanza el tiempo determinado anteriormente se procede a realizar la captura de una muestra de control.
- **Default.** Cuando la placa se reinicia de manera inesperada.

Tras esto ya tendríamos todos los elementos necesarios para controlar el modo Deep Sleep. Al utilizar este modo en nuestra placa conseguimos prolongar la duración de las baterías de una forma considerable.

### 5.3.10. Recogida de datos en la ESP8266

Como hemos explicado anteriormente, nuestra placa ESP8266 actúa como esclava de la ESP32. Esto se traduce en una lógica más simplificada, puesto que el número de funciones necesarias se reduce considerablemente.

El esquema general de funcionamiento es el siguiente:

- En la primera fase se establece conexión con la red Wi-Fi, el servidor MQTT y la plataforma para la visualización de datos Thinger.io, que se introducirá en secciones posteriores. Todo ello se realiza en el `setup()`.
- A continuación entramos en el bucle principal `loop()`. Dentro de este bucle el sistema empezará a recoger y enviar los datos del sensor de gas hasta que la placa ESP32 le avise mediante una interrupción en el pin designado para tal fin.
- En la última fase, realizaremos la desconexión de todos los servicios, y pondremos la placa en modo Deep Sleep. La placa permanecerá en este modo hasta que reciba en el pin `rst` una señal para que se reinicie y vuelva a comenzar la recogida de datos.

Dado que el tratamiento y envío de los datos es equivalente en ambas placas, aquí únicamente explicaremos cómo se produce la recogida de los gases y las pequeñas diferencias que existen en el modo Deep Sleep y la gestión de la memoria.

#### Gestión de la memoria

Gestionar la memoria de la placa ESP8266 no es una tarea tan trivial como en el caso de la ESP32. Esto es debido a que la librería `Preferences` no se encuentra disponible, y por tanto deberemos utilizar la librería `EEPROM`.

Para utilizar esta librería deberemos tener en cuenta el tamaño del elemento que queremos escribir. Esto se debe a que los elementos usualmente se almacenan de manera secuencial para optimizar el espacio, y es necesario conocer la dirección de memoria para poder realizar una lectura posterior de los datos almacenados.

Para mostrar el funcionamiento básico, hemos creado un supuesto donde tenemos dos elementos almacenados en la memoria, Figura 50.



Figura 50: Distribución de la memoria.

Para acceder al elemento A deberemos realizar una lectura en la posición 0, ya que los datos del elemento en cuestión empiezan ahí. Si quisiéramos obtener el elemento B, deberías de utilizar `sizeof()` con el tipo de dato del elemento A, para obtener dónde empieza el elemento B. Como podemos ver, deberemos de tener cuidado a la hora de acceder correctamente a las direcciones de memoria.

### Sensor MQ2 y MQ135

Para implementar los sensores y realizar las pruebas de gases, hemos utilizado dos aproximaciones diferentes. La primera está basada en la utilización de la librería *MQSensorsLib* [13], para la obtención de las concentraciones de los diferentes gases presentes en el aire. La segunda aproximación consiste en obtener los valores en raw directamente del sensor.

El proceso para obtener los valores que nos proporciona el MQ2 es realmente sencillo, pero para que el sensor ofrezca valores con la suficiente precisión es necesario un tiempo de precalentamiento que puede oscilar entre unos minutos y unas horas. Para realizar una lectura en raw de los valores, no es necesario aplicar ningún tipo de fórmula, ni parámetros específicos. Para obtener los valores hemos utilizado la función `AnalogRead()`, indicándole el pin en que está conectada la salida analógica del sensor. Debido a que la placa ESP8266 dispone de una resolución de 10 bits, el voltaje que nos proporciona el sensor como resultado estará expresado como un entero comprendido entre 0 y 1024. Si queremos obtener el número de ppm (partes

por millón) existentes, realizaremos una simple conversión, teniendo en cuenta que el mínimo que puede detectar está en las 300 ppm, y que el máximo se sitúa en las 10000 ppm.

Respecto al número de muestras que recoge el sensor, hemos estimado que la recogida de valores en intervalos de un segundo es lo más apropiado. Sin embargo, este parámetro puede ser modificado en el archivo de configuración correspondiente.

Respecto al esquema de conexiones, Figura 51, debemos de tener en cuenta que el sensor dispone de 4 pines, pero el software utilizado para la construcción de los diagrama incluye solamente un modelo que consta de 3. El pin restante se corresponde con la salida digital del sensor, que para este proyecto no ha sido necesario utilizarla.

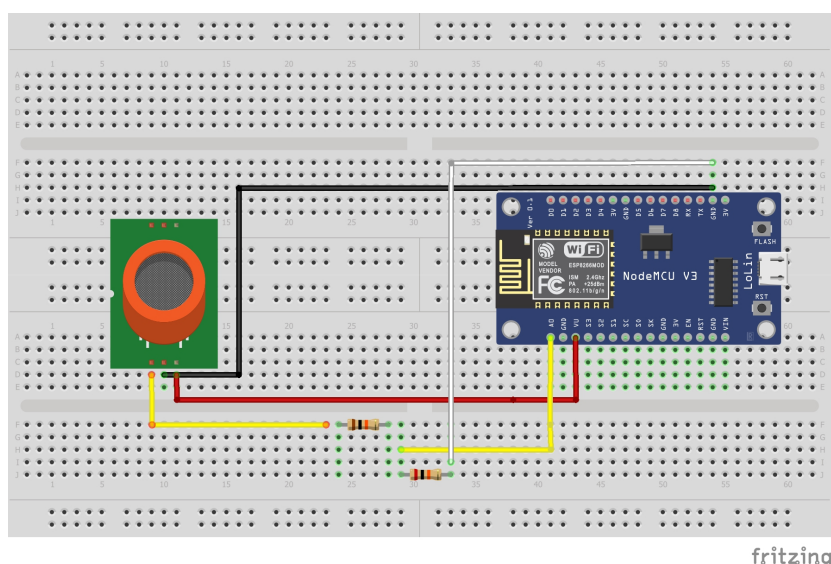


Figura 51: Esquema de conexiones para el sensor MQ2.

### 5.3.11. Recogidas de datos programadas

Ya hemos explicado como el sistema se inicia bajo las acciones directas de la persona a monitorizar, pero también hemos implementado otro modo de funcionamiento alternativo.

Este modo se inicia cuando se alcanza el tiempo previamente definido mientras la placa ESP32 se encuentre en suspensión o Deep Sleep. Al cumplirse esta condición

de tiempo se procede a tomar una imagen y los valores captados por el sensor de gas. Hemos decidido implementar este modo para poder controlar regularmente los valores en condiciones normales y en situaciones anómalas. Dichas condiciones anómalas pueden incluir la utilización de agentes limpiadores en el inodoro. Los gases persistentes después de la aplicación del limpiador podrían alterar significativamente los valores reales. Mediante la recogida de estas muestras podemos detectar este tipo de situaciones y evaluar los datos en función de ello.

De manera predefinida se recoge una muestra cada 60 segundos, pero este valor puede ser alterado en los archivos de configuración proporcionados.

## 5.4. Plataforma IoT

### 5.4.1. Definición

Una de las principales herramientas que han surgido a raíz del auge del Internet de las Cosas son las denominadas Plataformas IoT.

Las Plataformas IoT tienen como objetivo principal proporcionar a los desarrolladores una herramienta que simplifica el desarrollo integral de soluciones para el IoT. Entre las propiedades más relevantes que debe proporcionarnos se encuentran las siguientes:

- **Conectividad y normalización.** Poseen una serie de protocolos y formatos de datos en una interfaz software que garantiza la transmisión de los datos y la interacción con los dispositivos.
- **Gestión de los dispositivos.** Nos permiten visualizar el estado de la conexión de todos los dispositivos.
- **Base de datos.** Nos proporcionan un almacenamiento escalable para los datos generados por los dispositivos, permitiéndonos despreocuparnos completamente de la gestión del almacenamiento de los datos.
- **Procesamiento y gestión de eventos.** Permiten configurar acciones que se activen automáticamente en función de los datos de los sensores.

- **Análisis.** Pueden llevar a cabo análisis complejos de los datos básicos y aplicar técnicas de aprendizaje automático.
- **Visualización.** Nos proporcionan herramientas de visualización de los datos en forma de gráficos.
- **Interfaces externas.** Disponen de una serie de APIs y SDKs que nos permiten conectarla con sistemas de terceros.

#### 5.4.2. Plataformas IoT más relevantes

En esta sección vamos a ver las principales Plataformas IoT que existen.

##### **Node-RED**

Node-RED es una herramienta de programación visual programada en NodeJS y que puede desplegarse en dispositivos con baja potencia como una Raspberry Pi. Muestra de manera visual las diferentes relaciones y funciones, y permite al usuario añadir nuevas funcionalidades sin la necesidad de escribir código. Fue creada por Dave Conway-Jones y Nick O’Leary del grupo de Servicios de Tecnologías Emergentes de IBM en el año 2013.

La estructura mínima son los nodos. Estos se seleccionan a través de una interfaz gráfica y nos permiten realizar una tarea en concreto, como puede ser recibir un mensaje mediante el protocolo MQTT o programar el encendido de una luz. Dichos nodos se organizan en una estructura denominada flujo o flow.

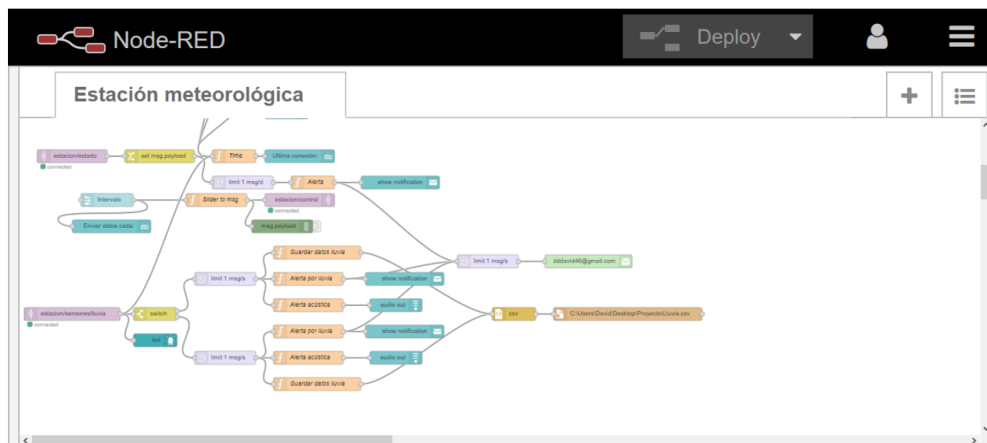


Figura 52: Ejemplo de un flow en Node-RED

## Thinger

Esta plataforma nos proporciona una nube escalable para conectar nuestros dispositivos. Es compatible con las placas de desarrollo de Arduino y las basadas en el chip ESP32 y ESP8266. Ofrece un panel de control desde el que podremos gestionar los diferentes servicios que nos ofrece, como pueden ser la creación de dashboards, estadísticas, dispositivos o data buckets. También se puede integrar con otros servicios como IFTTT (If This, Then That) para automatizar tareas.

Dispone de una versión gratuita que nos permitirá añadir hasta dos dispositivos y crear varios data buckets, pero de ser necesario añadir más dispositivos pone a nuestra disposición una serie de tarifas de pago en las que se incrementan las funcionalidades en función de las necesidades de nuestro proyecto.

## Zetta

Es una plataforma de IoT para la construcción de APIs basada en NodeJS. Zetta combina API REST y WebSockets para crear aplicaciones de datos intensivos y en tiempo real. Es una plataforma Open Source, por lo que podremos utilizarla sin ningún coste añadido.



Sus desventajas radican en la necesidad de poseer conocimientos de programación en NodeJS y la carencia de un soporte dedicado, con lo que tendremos que resolver las dudas mediante la comunidad de usuarios.

## Watson IoT Platform

Esta plataforma desarrollada por IBM [65], apuesta fuerte por el IoT, ofreciendo una gran cantidad de servicios. Proporciona un panel de control web desde el que podremos gestionar todos los servicios y dispositivos. Sus características principales son las siguientes:

- **Almacenamiento de datos.** Podemos almacenar los datos durante un periodo de tiempo.
- **Gestión de dispositivos.** Nos permite controlar nuestros dispositivos y obtener datos de diagnóstico.
- **MQTT.** Utiliza el protocolo MQTT para establecer las conexiones. Además, podremos utilizar el MQTT con TLS para proteger las comunicaciones.

## ThingSpeak

ThingSpeak es una plataforma de aplicaciones diseñada para conectar personas con objetos. Pone a nuestra disposición una API para almacenar y recuperar datos de los objetos utilizando el protocolo HTTP. Está basada en Ruby on Rails 3.0 (RoR), un framework para aplicaciones web de código abierto, con una arquitectura de Modelo-Vista-Controlador.

Esta plataforma es muy completa, ofreciéndonos acceso a una serie de funciones como ThingTweet, ThingHTTP, TweetControl, TalkBack o TimeControl, que nos permitirán desde publicar tweets hasta ejecutar tareas a una hora determinada. Dispone de varias versiones en función del tipo de proyecto que queramos realizar, ofreciendo una versión gratuita lo suficientemente completa como para realizar pruebas.

### 5.4.3. Elección de la Plataforma IoT

Como hemos podido ver, las Plataformas IoT disponen de una serie de funciones que son muy interesantes para nuestro proyecto. Principalmente vamos a utilizar las capacidades de gestión de los dispositivos y la visualización de los datos, por lo que sumado al hecho de que solo contamos con dos dispositivos hemos decidido utilizar Thinger como plataforma IoT.

Para conectar Thinger con nuestros dispositivos ha sido necesario realizar una serie de pasos:

- Deberemos definir los dispositivos que queremos conectar, mediante la creación de sus respectivas credenciales de acceso.
- Crearemos una serie de Data Buckets donde almacenaremos toda la información que queramos posteriormente visualizar.
- Creamos un dashboard. Aquí ubicaremos todas las gráficas que necesitemos. Los datos se obtienen directamente de los Data Buckets previamente definidos.

Para este proyecto se han definido tres Data Buckets, ToiletLidStatus, GasData y ThermalData. Estos se encargarán de registrar la apertura de la tapa, los valores registrados por el sensor de gas y las temperaturas obtenidas por la cámara térmica.

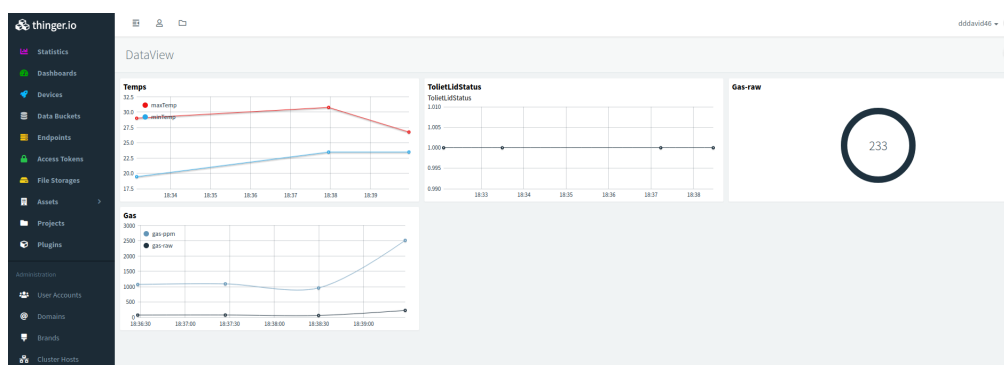


Figura 53: Dashboard de Thinger.io

Para construir nuestro dashboard hemos utilizado una serie de gráficas:

- **Temps:** Esta gráfica contiene las máximas y las mínimas temperaturas registradas por la cámara térmica en cada uso.
- **ToiletLidStatus:** En este diagrama se muestra el momento en que el habitante abre la tapa del inodoro. Debido a las limitaciones en las peticiones, es posible que no se registre el momento en que el habitante cierre la tapa.
- **Gas-raw:** En esta gráfica se muestran las lecturas en raw captadas por el sensor de gas. Muestra únicamente el último registro.
- **Gas:** Este diagrama muestra las ppm (partes por millón) y los valores raw registrados por el sensor de gas.

Debemos de tener en cuenta que al utilizar una versión gratuita solamente podremos realizar una petición por minuto.

## 5.5. Visualización de los datos

Visualizar la información recogida por los sensores es un punto muy importante, ya que permite al usuario/cuidador monitorizar los datos de una manera sencilla. Los datos se pueden consultar desde la propia plataforma IoT o a través de una aplicación que permita conectarse a la base de datos mediante una interfaz gráfica.

A continuación se muestran las gráficas que se han diseñado en la plataforma IoT para visualizar los datos.

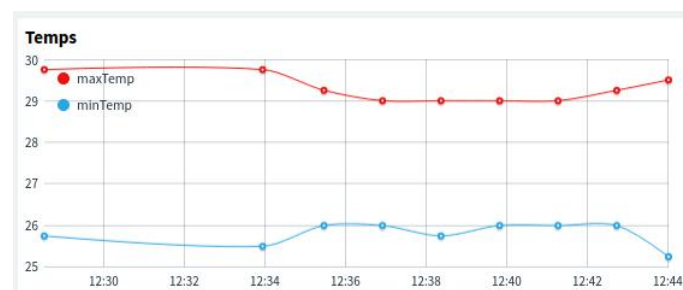


Figura 54: Histórico de temperaturas máximas y mínimas registradas por la cámara térmica.



Figura 55: Registro de uso del inodoro.



Figura 56: Último valor raw obtenido por el sensor de gas.

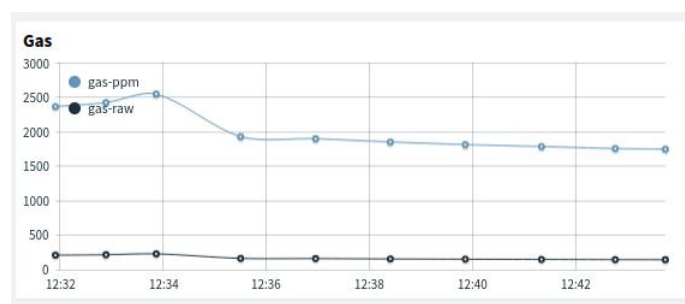


Figura 57: Histórico de valores recogidos por el sensor de gas.

También es posible consultar la información directamente de los Data Buckets previamente creados, y consultar datos acerca de los dispositivos vinculados a la plataforma.

| Bucket Data              |         |         |
|--------------------------|---------|---------|
| Date                     | Gas-ppm | Gas-raw |
| 2021-08-25T12:43:43.981Z | 1749    | 153     |
| 2021-08-25T12:42:45.763Z | 1758    | 154     |
| 2021-08-25T12:41:19.928Z | 1787    | 157     |
| 2021-08-25T12:39:51.336Z | 1815    | 160     |
| 2021-08-25T12:38:23.997Z | 1853    | 164     |
| 2021-08-25T12:36:57.140Z | 1900    | 169     |
| 2021-08-25T12:35:29.925Z | 1929    | 172     |
| 2021-08-25T12:33:51.715Z | 2545    | 237     |
| 2021-08-25T12:32:53.338Z | 2421    | 224     |
| 2021-08-25T12:31:54.533Z | 2365    | 218     |
| 2021-07-04T18:39:28.157Z | 2507    | 233     |
| 2021-07-04T18:38:29.844Z | 963     | 70      |
| 2021-07-04T18:37:26.987Z | 1095    | 84      |
| 2021-07-04T18:36:28.010Z | 1076    | 82      |
| 2021-06-16T15:15:09.211Z | 399     | 145     |
| 2021-06-16T15:12:42.589Z | 407     | 157     |

Figura 58: Data Bucket del sensor de gas.

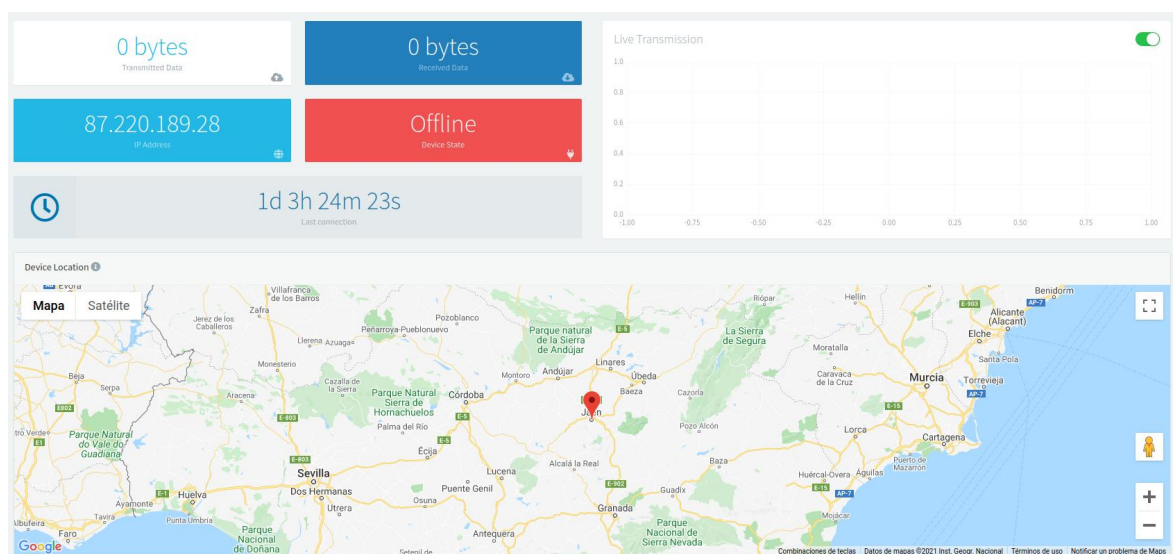


Figura 59: Información del dispositivo.

Si se requiere acceder a todos los datos recogidos, es posible realizarlo mediante la aplicación MongoDB Compass, que permite consultar la base de datos mediante una interfaz gráfica.

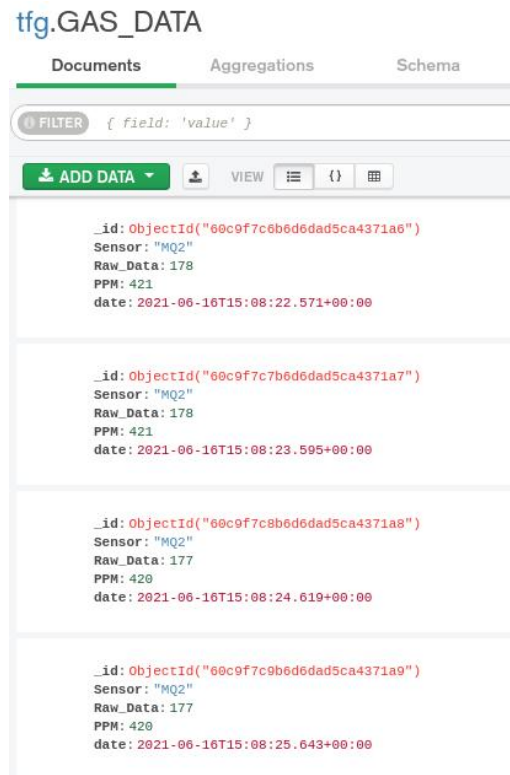


Figura 60: Información de la base de datos.

## 5.6. Dispositivo instalado

En esta sección se mostrará el dispositivo ya instalado en el inodoro.



Figura 61: Dispositivo instalado en el inodoro.



Figura 62: Sistema de alimentación

## 6. Validación y evaluación

En este punto de la memoria explicaremos la prueba que hemos realizado para verificar que el producto final cumple con los requisitos previamente definidos. Dos pruebas en las que compararemos la variación que se produce en los diferentes tipos de gases, mediante el uso de las librerías específicas de los sensores MQ2 y MQ135.

### 6.1. Pruebas del sistema

#### 6.1.1. Prueba de funcionamiento

Aunque durante el desarrollo del proyecto se han realizado múltiples pruebas de funcionamiento para comprobar que cada nueva funcionalidad introducida se comportaba de manera adecuada, es necesario realizar una prueba del conjunto completo, para verificar que el sistema funciona adecuadamente.

#### 6.1.2. Prueba con el sensor MQ2

Para esta prueba utilizaremos una adaptación de la librería MQSensorsLib [13] que nos permite obtener simultáneamente las ppm de los distintos gases.

Para usar esta librería deberemos de configurar una serie de parámetros:

- **Voltage\_Resolution.** Este valor se corresponde con el máximo voltaje de entrada en nuestra placa de desarrollo. En el caso de la ESP8266 es 3.3V.
- **RatioMQ2CleanAir.** Este valor nos lo proporciona la librería en función del tipo de sensor, 9.83 .
- **R0.** Es la resistencia del sensor en aire limpio. La librería obtiene este valor realizando una media de los valores recogidos durante un periodo de tiempo.
- **RL.** Es la resistencia de carga. Su valor viene serigrafiado en la resistencia del sensor MQ. En nuestro caso viene con una resistencia de  $1k\Omega$ , pero es aconsejable sustituirla por una de  $20k\Omega$  para obtener lecturas más precisas.



### 6.1.3. Prueba con el sensor MQ135

Al igual que con el sensor MQ2 utilizaremos una adaptación de la librería MQSensorsLib [13] que nos permite obtener simultáneamente las ppm de los distintos gases.

Los parámetros necesarios son los siguientes:

- **Voltage\_Resolution**.3.3V.
- **RatioMQ2CleanAir**. 3.6 .
- **R0**. Obtenida automáticamente por la librería.
- **RL**. 1k $\Omega$ , es aconsejable sustituirla por una de 20k $\Omega$  .

## 6.2. Pruebas del sensor en reposo y en activo

Esta prueba tiene como objetivo comparar el comportamiento del sensor mientras el sensor se encuentra en estado de reposo (muestras recogidas cada 60 segundos sin deposiciones presentes) y con el sensor en un estado activo (el habitante está realizando las deposiciones). Además, no se empleará el uso de la librerías específicas, es decir, los valores estarán en raw.

## 6.3. Resultados

A continuación expondremos una serie de valores obtenidos durante las distintas pruebas.

### 6.3.1. Sistema

El sistema se comporta en general de manera adecuada con los parámetros definidos por defecto, sin embargo, si se disminuye el tiempo entre la toma de muestras el broker MQTT experimenta ralentizaciones. También se ha detectado que de manera aleatoria se produce una pérdida de conexión inalámbrica en la placa ESP32. En principio esto se debe a un error presente en las propias librerías utilizadas para establecer la conexión.

**6.3.2. MQ2**

Los resultados aquí expuestos se corresponden con los datos de dos muestras antes de las deposiciones y tres durante las mismas.

| Alcohol | C0   | H2  | LPG | Propane |
|---------|------|-----|-----|---------|
| 9.8     | 38.5 | 8.3 | 4.2 | 5.4     |
| 8.6     | 33.1 | 7.5 | 3.8 | 4.9     |
| 8.1     | 30.9 | 7.1 | 3.6 | 4.7     |
| 7.5     | 28.4 | 6.7 | 3.4 | 4.4     |
| 6.5     | 23.8 | 5.9 | 3.0 | 3.9     |
| 4.5     | 15.5 | 4.4 | 2.2 | 2.9     |
| 4.6     | 16.3 | 4.5 | 2.2 | 3.0     |
| 4.3     | 15.0 | 4.3 | 2.1 | 2.8     |
| 4.5     | 15.5 | 4.4 | 2.2 | 2.9     |
| 4.9     | 17.4 | 4.8 | 2.4 | 3.1     |

Tabla 37: Resultado de la prueba de gases con el sensor MQ2.

**6.3.3. MQ135**

Los resultados aquí expuestos se corresponden con los datos de dos muestras antes de las deposiciones y tres durante las mismas.

| Alcohol | C02 | CO  | Tolueno | NH4 | Acetona |
|---------|-----|-----|---------|-----|---------|
| 0.9     | 2.0 | 2.4 | 0.3     | 3.2 | 0.3     |
| 0.9     | 2.1 | 2.6 | 0.3     | 3.3 | 0.3     |
| 1.1     | 2.3 | 3.1 | 0.4     | 3.7 | 0.3     |
| 1.1     | 2.5 | 3.3 | 0.4     | 3.8 | 0.4     |
| 1.2     | 2.6 | 3.4 | 0.5     | 3.9 | 0.4     |
| 1.1     | 2.4 | 3.4 | 0.5     | 3.9 | 0.3     |
| 1.2     | 2.4 | 3.6 | 0.5     | 3.8 | 0.4     |
| 1.8     | 2.5 | 3.4 | 0.5     | 3.9 | 0.4     |
| 1.1     | 2.5 | 3.3 | 0.5     | 3.9 | 0.4     |

Tabla 38: Resultado de la prueba de gases con el sensor MQ135

Como se ha podido ver, aunque la prueba de gases específicos difiere en las sustancias a las que son reactivos cada uno de los sensores, se ha seleccionado como sensor a utilizar en el prototipo final el sensor MQ2. Esto es debido a que los resultados muestran una mayor concentración a las sustancias que el sensor MQ135.

#### 6.3.4. Sensor en reposo y en activo

| Raw |
|-----|
| 163 |
| 187 |
| 191 |
| 175 |
| 189 |
| 178 |
| 186 |
| 182 |
| 179 |
| 185 |

Tabla 39: Datos obtenidos en reposo por el sensor MQ2

Como se puede observar, el valor obtenido por el sensor suele fluctuar en el rango de 160-190.

| Raw |
|-----|
| 338 |
| 362 |
| 287 |
| 324 |
| 345 |
| 359 |
| 295 |
| 333 |
| 356 |
| 358 |

Tabla 40: Datos obtenidos en reposo por el sensor MQ2

Como se puede observar, los valores cuando el sensor se encuentra activo fluctúan en el rango de 280-370. Por tanto, existe una diferencia cuando el sensor se encuentra en un estado de reposo y activo, lo que nos indica que el dispositivo construido es capaz de detectar los gases derivados de las deposiciones.

## **7. Conclusiones**

### **7.1. Revisión del conocimiento adquirido**

En este proyecto se han expuesto las diferentes fases o tareas para el desarrollo de un sistema IoT destinado a monitorizar los gases y temperatura de las deposiciones. Entre ellas, la selección del hardware para el dispositivo, la programación de las placas de desarrollo y los sensores, la implementación de los distintos protocolos necesarios para realizar la comunicación, y el uso de una plataforma IoT para la gestión y visualización de los datos. Este proyecto también me ha permitido introducirme en el mundo de las bases de datos NoSQL, permitiendo conocer sus ventajas y desventajas respecto a las bases de datos relacionales en una aplicación real. Este proyecto también ha servido para adquirir conocimientos fuera de mi disciplina, como puede ser el diseño asistido por computadora o CAD.

### **7.2. Trabajo futuro**

Aunque la aplicación y el prototipo satisfacen los requisitos propuestos inicialmente en el proyecto, esto no quiere decir que no estén sujetos a cambios o mejoras. Entre dichas mejoras se encontrarían las siguientes:

1. Mejoras en la gestión de hilos.

Actualmente el sistema de gestión de los mensajes entrantes se gestiona únicamente en un hilo de procesamiento. Para la carga actual de mensajes el sistema se comporta adecuadamente, pero si aumentamos el número de muestras por segundo el sistema se resiente.

2. Diseño del sistema energético.

El prototipo actual está alimentado mediante una batería o powerbank de dimensiones destacables. Para facilitar su integración en los entornos reales se podría optar por un sistema de baterías de pequeñas dimensiones ubicadas en el interior

del módulo principal. Esto disminuiría el número de cables exteriores del dispositivo, permitiendo un mayor grado de integración en entornos reales.

### 3. Inclusión de sensores.

El prototipo actual dispone de un único sensor de gas situado en la parte posterior de la tapa. Sería interesante añadir un sensor de gas en la parte frontal de la tapa con el fin de recopilar muestras adicionales y contrastar los datos obtenidos.

### 4. Sustitución de la resistencia de carga.

Según la documentación técnica asociada a los sensores, se recomienda utilizar una resistencia de carga de  $20k\Omega$  , en lugar de la de  $1k\Omega$  con la que viene equipada el sensor.

### 5. Ensamblaje del dispositivo.

Actualmente las distintas placas y sensores que conforman el dispositivo están integradas sobre una breadboard o placa de pruebas. Esto es adecuado para entornos de desarrollo, puesto que facilita la conexión de los diferentes elementos que conforman el dispositivo, pero en un entorno real suponen un aumento en el volumen total del dispositivo, perjudicando su capacidad de integración.

Una manera de reducir el volumen total del dispositivo consistiría en sustituir la placa de pruebas por un circuito impreso. Este tipo de circuitos consisten en una lámina delgada de silicio y cobre, que contiene una serie de orificios para insertar los componentes, permitiendo realizar la soldadura de los componentes y como consecuencia una disminución del volumen.

#### 6. Ampliación de la experimentación.

Es necesario la realización de más experimentos para corroborar que este dispositivo funciona. Para ello se deberían recabar muestras de dos grupos poblacionales distintos, uno de ellos formado por personas sanas y otro conformado por personas con cáncer.

#### 7. Aplicación de técnicas de aprendizaje supervisado o no supervisado.

La aplicación de este tipo de técnicas podría ayudarnos a clasificar los datos de manera automática. Para ello es necesario que nuestro dispositivo sea capaz de discernir entre una persona sana y una persona con cáncer, con el fin de alimentar mediante estos datos a un modelo para el aprendizaje supervisado.

#### 8. Plataforma de visualización.

Como hemos comentado con anterioridad, la plataforma empleada ofrece una serie de limitaciones en su versión gratuita. Se podría sustituir por otra plataforma que no tenga dichas limitaciones u optar por adquirir un plan de pago.



## 8. Bibliografía

### Referencias

- [1] URL: <https://www.instintoprogramador.com.mx/2018/04/modelo-incremental.html>.
- [2] 1.8.2021. URL: <http://controlatuprogreso.es/mejores-pulseras-de-actividad/>.
- [3] 1.8.2021. URL: <https://ultimaker.com/es/software/ultimaker-cura>.
- [4] 1.8.2021. URL: <https://www.cnet.com/reviews/nest-thermostat-e-review/>.
- [5] 1.8.2021. URL: <https://pubsubclient.knolleary.net/>.
- [6] *About Node-RED*. 25.6.2021. URL: <https://nodered.org/about/>.
- [7] *Activity Diagrams*. URL: <https://www.uml-diagrams.org/activity-diagrams.html>.
- [8] *Adafruit MMA8451 Library*. 4.3.2021. URL: [https://github.com/adafruit/Adafruit\\_MMA8451\\_Library](https://github.com/adafruit/Adafruit_MMA8451_Library).
- [9] S. Bennet et al. *Análisis y Diseño Orientado a Objetos de Sistemas Usando UML. 3a Ed.* McGraw-Hill, 2007.
- [10] *Apache ActiveMQ*. URL: <https://activemq.apache.org/>.
- [11] K. Beck y col. "Manifesto for Agile Software Development". En: 2013.
- [12] Benoît Blanchon. *ArduinoJson*. 20.6.2021. URL: <https://arduinojson.org/>.
- [13] Miguel Angel Califa Urquiza, Ghiordy Contreras Contreras y Yerson Ramiro Carrillo Amado. *miguel5612/MQsensorsLib: Arduino Preview V1.03*. Sep. de 2019. DOI: 10.5281/zenodo.3384301. URL: <https://doi.org/10.5281/zenodo.3384301>.
- [14] Yuang Chen y Thomas Kunz. "Performance evaluation of IoT protocols under a constrained wireless access network". En: *2016 International Conference on Selected Topics in Mobile & Wireless Networking (MoWNeT)*. IEEE. 2016, págs. 1-7.

- [15] Cisco. *What Is Edge Computing?* 22.6.2021. URL: <https://www.cisco.com/c/en/us/solutions/computing/what-is-edge-computing.html>.
- [16] *CoAP RFC 7252 Constrained Application Protocol*. URL: [https://www.oasis-open.org/committees/tc\\_home.php?wg\\_abbrev=amqp](https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=amqp).
- [17] *Computación en el borde*. 24.6.2021. URL: [https://es.wikipedia.org/wiki/Edge\\_computing](https://es.wikipedia.org/wiki/Edge_computing).
- [18] *Computación en la niebla*. 24.6.2021. URL: [https://en.wikipedia.org/wiki/Fog\\_computing](https://en.wikipedia.org/wiki/Fog_computing).
- [19] *El modelo en cascada: desarrollo secuencial de software*. 22.5.2021. URL: <https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/el-modelo-en-cascada/>.
- [20] *Erlang MQTT Broker*. URL: <https://emqtt.io/>.
- [21] *ESP32 PWM with Arduino IDE (Analog Output)*. 20.6.2021. URL: <https://randomnerdtutorials.com/esp32-pwm-arduino-ide/>.
- [22] Alejandro Frechina. *Metodología Scrum ¿Que es?* 20.6.2021. URL: <https://winred.es/management/metodologia-scrum-que-es/gmx-niv116-con24594.htm>.
- [23] Rikke Gade y Thomas B Moeslund. "Thermal cameras and applications: a survey". En: *Machine vision and applications* 25.1 (2014), págs. 245-262.
- [24] Henry Laurence Gantt. *Work, wages, and profits*. Engineering Magazine Company, 1913.
- [25] *Getting Started with VS Code and PlatformIO IDE for ESP32 and ESP8266 (Windows, Mac OS X, Linux Ubuntu)*. 20.6.2021. URL: <https://randomnerdtutorials.com/vs-code-platformio-ide-esp32-esp8266-arduino/>.
- [26] Geoffrey Gregis y col. "Detection and quantification of lung cancer biomarkers by a micro-analytical device using a single metal oxide-based gas sensor". En: *Sensors and Actuators B: Chemical* 255 (2018). 20.6.2021, págs. 391-400. ISSN: 0925-4005. DOI: <https://doi.org/10.1016/j.snb.2017.08.056>. URL: <https://www.sciencedirect.com/science/article/pii/S0925400517314788>.

- [27] Urs Hunkeler, Hong Linh Truong y Andy Stanford-Clark. "MQTT-S—A publish/subscribe protocol for Wireless Sensor Networks". En: *2008 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE'08)*. IEEE. 2008, págs. 791-798.
- [28] IBM. *Cloud Computing*. 22.6.2021. URL: <https://www.ibm.com/cloud/learn/cloud-computing>.
- [29] Espressif Inc. *ESP32 Technical Reference*.
- [30] Espressif Inc. *ESP8266 Technical Reference*.
- [31] *Insight Into ESP32 Sleep Modes Their Power Consumption*. 28.6.2021. URL: <https://lastminuteengineers.com/esp32-sleep-modes-power-consumption/>.
- [32] Mariusz Kaczmarek y Antoni Nowakowski. "Active IR-thermal imaging in medicine". En: *Journal of Nondestructive Evaluation* 35.1 (2016), pág. 19.
- [33] Jeff Sutherland Ken Schwaber. "The 2020 Scrum Guide". En: (2020).
- [34] BB Lahiri y col. "Medical applications of infrared thermography: a review". En: *Infrared Physics & Technology* 55.4 (2012), págs. 221-235.
- [35] Craig Larman y Victor R Basili. "Iterative and incremental developments. a brief history". En: *Computer* 36.6 (2003), págs. 47-56.
- [36] Patricio Letelier y M<sup>a</sup> Carmen Penadés. "Metodologías ágiles para el desarrollo de software: eXtreme Programming (XP)". En: (2006).
- [37] Cesare Malagù y col. "Chemoresistive Gas Sensors for the Detection of Colorectal Cancer Biomarkers". En: *Sensors* 14.10 (2014), págs. 18982-18992. ISSN: 1424-8220. DOI: 10.3390/s141018982. URL: <https://www.mdpi.com/1424-8220/14/10/18982>.
- [38] Félix Marco Millán. "Diseño e implementación de un sistema de medida de gases con Arduino". En: (2016).
- [39] *MQ-2 Semiconductor Sensor for Combustible Gas*. 6.4.2021.
- [40] *MQTT: The Standard for IoT Messaging*. 23.6.2021. URL: <https://mqtt.org/>.

- [41] Nitin Naik. "Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP". En: *2017 IEEE international systems engineering symposium (ISSE)*. IEEE. 2017, págs. 1-7.
- [42] *OASIS Advanced Message Queuing Protocol (AMQP)*. URL: <https://coap.technology/>.
- [43] Mike O Ojo y col. "A review of low-end, middle-end, and high-end IoT devices". En: *IEEE Access* 6 (2018), págs. 70528-70554.
- [44] Dinkar R Patnaik Patnaikuni. "A Comparative Study of Arduino, Raspberry Pi and ESP8266 as IoT Development Board." En: *International Journal of Advanced Research in Computer Science* 8.5 (2017).
- [45] *PlatformIO IDE — PlatformIO 5.2.0a3 documentation*. 6.4.2021. 2014. URL: <https://docs.platformio.org/en/latest/integration/ide/pioide.html>.
- [46] Zhi-Hong Qian y Yi-jun Wang. "IoT technology and application". En: *Acta Electronica Sinica* 40.5 (2012), págs. 1023-1029.
- [47] R.Pressman. *Software Engineering. A Practitioner's Approach. 8th Ed.* McGraw-Hill, 2014.
- [48] *RASPBERRY PI 3 MODEL B+*. 27.5.2021. URL: <https://www.distrelec.de/en/raspberry-pi-model-1gb-ram-raspberry-pi-raspberry-pi-model-bulk/p/30109158>.
- [49] ResearchGate. "Wireless body sensor networks for health-monitoring applications - Scientific Figure on ResearchGate". En: (). 20.6.2021. URL: [https://www.researchgate.net/figure/ZigBee-network-topologies-51\\_fig2\\_23307553](https://www.researchgate.net/figure/ZigBee-network-topologies-51_fig2_23307553).
- [50] Kinnera Bharath Kumar Sai, Subhaditya Mukherjee y H Parveen Sultana. "Low Cost IoT based air quality monitoring setup using arduino and MQ series sensors with dataset analysis". En: *Procedia Computer Science* 165 (2019), págs. 322-327.
- [51] S Santos. *ESP32 Deep Sleep with Arduino IDE and Wake Up Sources. Random Nerd Tutorials*. 20.6.2021. URL: <https://randomnerdtutorials.com/esp32-deep-sleep-arduino-ide-wake-up-sources/>.

- [52] Ken Schwaber. "Scrum development process". En: *Business object design and implementation*. Springer, 1997, págs. 117-134.
- [53] *Simple Text Oriented Messaging Protocol*. URL: <https://stomp.github.io/>.
- [54] *Sueldo de un analista programador*. 6.4.2021. URL: <https://es.indeed.com/career/salaries/Analista%20>.
- [55] *Sueldo de un jefe de proyectos*. 6.4.2021. URL: <https://es.indeed.com/career/jefe-de-proyecto/salaries>.
- [56] *Sueldo de un programador junior*. 6.4.2021. URL: <https://es.indeed.com/career/programador-senior/salaries>.
- [57] *Sueldo de un programador senior*. 6.4.2021. URL: <https://es.indeed.com/career/programador-senior/salaries>.
- [58] The FreeCAD Team. *FreeCAD*. 20.6.2021. URL: <https://www.freecadweb.org/>.
- [59] *TECHNICAL DATA MQ-135 GAS SENSOR*. 20.6.2021. URL: <https://www.olimex.com/Products/Components/Sensors/Gas/SNS-MQ135/resources/SNS-MQ135.pdf>.
- [60] *TECHNICAL DATA MQ-2 GAS SENSOR*. 6.4.2021.
- [61] *The world of IoT computing — Cloud vs Edge vs Fog*. 24.6.2021. URL: <https://avirup171.medium.com/the-world-of-iot-computing-cloud-vs-edge-vs-fog-28670e9be83d>.
- [62] Kimberly Tuck. *Motion and Freefall Detection Using the MMA8451, 2, 3Q*.
- [63] *UML Class Diagrams Overview*. URL: <https://www.uml-diagrams.org/class-diagrams-overview.html>.
- [64] *UML Use Case Diagrams*. URL: <https://www.uml-diagrams.org/use-case-diagrams.html>.
- [65] *Watson IoT Platform*. 14.6.2021. URL: <https://www.ibm.com/es-es/cloud/watson-iot-platform>.
- [66] *What is MQTT in IoT*. 22.6.2021. URL: <https://www.rfwireless-world.com/Terminology/MQTT-protocol.html>.
- [67] *Wi-Fi Alliance*. 1.8.2021. URL: <https://www.wi-fi.org/>.

- [68] *Zigbee*. 22.6.2021. URL: <https://zigbeealliance.org/es/soluci%C3%B3n/Zigbee/>.

## 9. Anexo

### 9.1. Manual de instalación

En esta sección se explicarán los requisitos previos y los pasos necesarios para desplegar el dispositivo.

El primer paso que deberemos realizar será descargar el software Visual Studio Code desde la página oficial: <https://code.visualstudio.com/>. Una vez descargado procederemos a instalarlo. Para poder subir el software a las placas de desarrollo será necesario añadir previamente dos extensiones. Estas extensiones se corresponden con la de C/C++ y la de PlatformIO IDE.

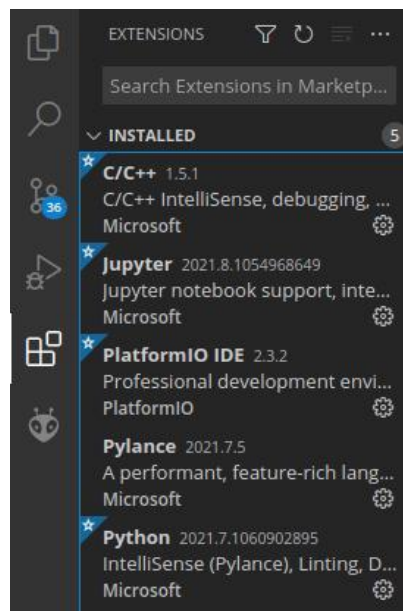


Figura 63: Panel lateral de Visual Studio Code.

Una vez hayamos añadido las extensiones correspondientes ya podemos proceder a subir el código correspondiente a nuestras placas de desarrollo. Para ello deberemos conectarlas mediante el puerto usb a nuestro ordenador y pulsar el botón de *upload* presente en la barra inferior de Visual Studio Code.



Figura 64: Barra inferior de VSCode. El botón de *upload* aparece resaltado.

Para poder conectarse a la red Wi-Fi correspondiente y a la plataforma IoT será necesario configurar una serie de parámetros. Para ello editaremos el sketch y cambiaremos los valores según corresponda.

```
//Parámetros de la conexión
const char SSID []= "MiFibra-8196-24G";
const char PASSWORD[] = "RXI5euREnwWpwauN2%4RvWJPm";
const int WIFI_TIME_OUT = 10000;
const char MQTT_SERVER[] = "192.168.1.33";
const int MQTT_PORT = 1883;

//Parámetros para Thinger.io
#define USERNAME "dddavid46"
#define DEVICE_ID "ESP32_Master"
#define DEVICE_CREDENTIAL "NMHMLR1SKsAxkU-d"
```

Figura 65: Parámetros a configurar.

La configuración del sistema del sumidero que a continuación se describirá, se ha diseñado para sistemas basados en Linux. Para que el script de Python correspondiente con el sistema sumidero se inicie automáticamente es necesario editar el archivo `/etc/profile`. En él incluiremos la siguiente línea: `sudo python3 /location/Mqtt.py &`. Debemos sustituir *location* por el directorio en el que hayamos situado nuestro script. El `&` es necesario puesto que nuestro script se ejecuta continuamente mientras el usuario no lo finalice.

Para la base de datos utilizaremos una imagen que nos genera un contenedor basado en mongo db. Para ello buscaremos la imagen oficial de docker en docker hub. Para instalarla abriremos un terminal y escribiremos el siguiente comando.

---

```
docker run -d -p 8050:8050 --name database mongo:5.0
```

---

El comando hace lo siguiente:

- **-d:** Permite que ejecutemos el proceso en background.
- **-p 8050:8050:** Las conexiones del puerto 8050 del contenedor se redireccionan al 8050 de nuestra máquina host.
- **--name database:** Indicamos el nombre de nuestro contenedor.



- **mongo:5.0:** Nombre de la imagen y versión.

Para poder crear la base de datos será necesario conectarse previamente a nuestro contenedor con el siguiente comando:

---

```
docker exec -it database bash
```

---

Una vez hecho esto, ya podremos acceder al shell de mongo mediante el comando *mongo*.

Para crear una base de datos utilizaremos el siguiente comando:

---

```
use tfg
```

---

Una vez hecho esto, crearemos una colección con el nombre deseado y un documento para que no se elimine automáticamente la colección.

---

```
db.THERMAL_DATA.insertOne({ name: 'test' })
```

---

También será necesario generar un usuario con permisos limitados para que se pueda consultar la información posteriormente.

---

```
use tfg
db.createUser(
{
  user: "user",
  pwd: passwordPrompt(),
  roles: [
    { role: "read", db: "tfg" }
  ]
}
)
```

---

Para desplegar el sistema sumidero es necesario descargar e instalar el broker Mosquitto <https://mosquitto.org/download/>. Una vez descargado e instalado, será necesario ejecutar el archivo Mqtt.py. Para que el programa se ejecute automáticamente en caso de reinicio del sistema será necesario editar un archivo (Linux solamente) mediante la siguiente orden.

---

```
sudo nano /etc/rc.local
```

---

A continuación, añadimos la siguiente línea antes del exit 0.

---

```
sudo python3 /home/Mqtt.py &
```

---

El & permite que el sistema continúe su inicio normal mientras el script se ejecuta.

La siguiente fase consistirá en configurar la plataforma IoT escogida, en este caso Thinger.io. Se deberá crear una cuenta nueva en la plataforma. A continuación, se iniciará sesión y se accederá al apartado Data Buckets. Dentro de este apartado se deberán crear los contenedores para los datos enviados por los dispositivos. La opción más importante es la de Data Source, que determina como se obtendrá la información. La estableceremos como From Device Write Call. Una vez se han creado los Data Buckets, se deberá crear un Dashboard. Para ello se accederá a la sección con el mismo nombre y se procederá a crearlo. Una vez creado, queda añadir los diferentes widgets. Estos widgets pueden obtener los datos directamente de las peticiones que realiza el dispositivo o de los Data Buckets que previamente se han creado.

A continuación, se deberá acceder a la sección Devices. Aquí se deberán añadir los dispositivos requeridos, estableciendo sus identificadores y credenciales, que serán necesarias para poder realizar la conexión con los dispositivos.

Una vez realizados estos pasos ya podemos proceder a instalar nuestro dispositivo. Para ello deberemos fijar los dos módulos a la tapa del inodoro, con la cámara y el sensor de gas apuntando hacia abajo. Una vez fijados, bajaremos la tapa del inodoro y procederemos a conectar los dispositivos a la batería.

## 9.2. Manual de usuario

En esta sección se explicará como un usuario/cuidador puede visualizar la información recogida por el sistema.

Para visualizar los datos desde la plataforma IoT se deberá acceder mediante las credenciales definidas previamente.

Una vez dentro de la plataforma IoT, los datos se pueden consultar desde la sección Dashboard.

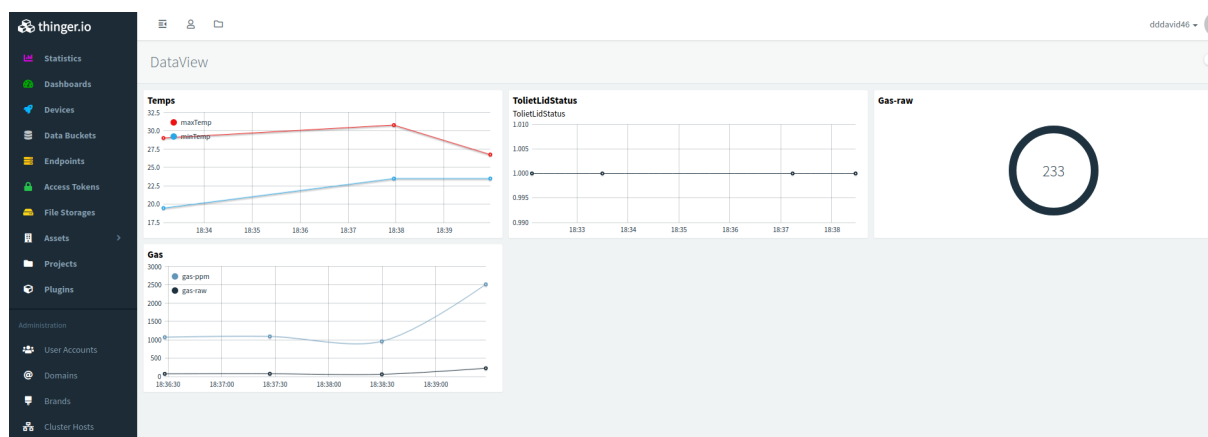


Figura 66: Dashboard de Thingier.io.

Desde la plataforma también se puede consultar el estado de los dispositivos ubicados en el inodoro. Para ello se accederá a la pestaña Devices y se seleccionará el dispositivo que se quiera consultar.

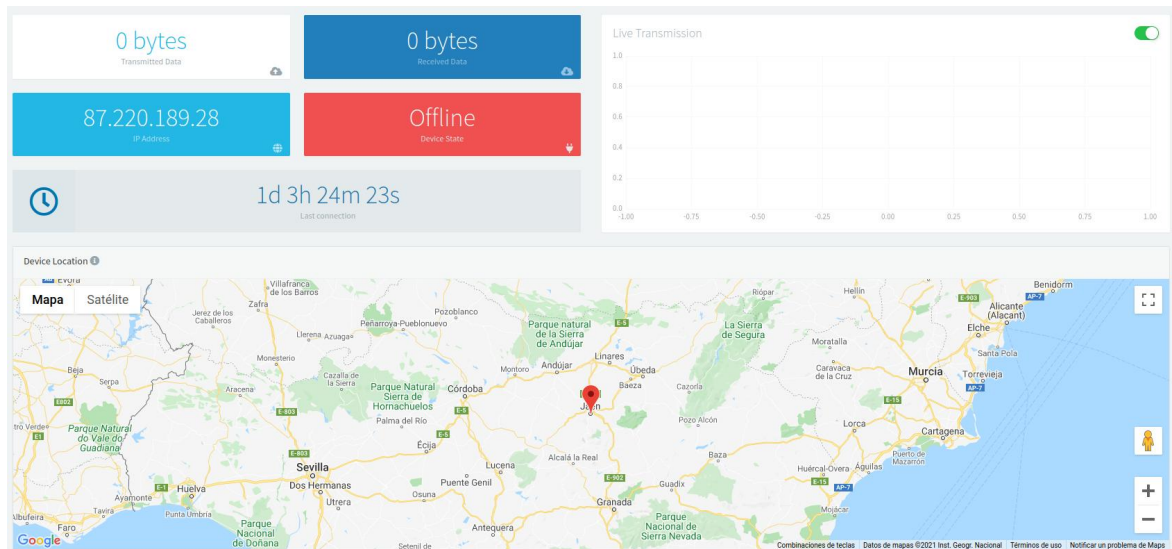


Figura 67: Información del dispositivo.

Adicionalmente, si se requiere consultar de manera más detallada la información recopilada por los sensores, se deberá acceder a la base de datos. Para ello será necesario descargar la aplicación MongoDB Compass, <https://www.mongodb.com/try/download/compass>.

Una vez instala la aplicación, será necesario introducir las credenciales definidas previamente, así como la dirección IP de la ubicación de la base de datos y el puerto.

tfg.GAS\_DATA

Documents Aggregations Schema

FILTER { field: 'value' }

ADD DATA VIEW

|                                                                                                                                                   |
|---------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre>{   "_id": "60c9f7c6b6d6dad5ca4371a6",   "Sensor": "MQ2",   "Raw_Data": 178,   "PPM": 421,   "date": "2021-06-16T15:08:22.571+00:00" }</pre> |
| <pre>{   "_id": "60c9f7c7b6d6dad5ca4371a7",   "Sensor": "MQ2",   "Raw_Data": 178,   "PPM": 421,   "date": "2021-06-16T15:08:23.595+00:00" }</pre> |
| <pre>{   "_id": "60c9f7c8b6d6dad5ca4371a8",   "Sensor": "MQ2",   "Raw_Data": 177,   "PPM": 420,   "date": "2021-06-16T15:08:24.619+00:00" }</pre> |
| <pre>{   "_id": "60c9f7c9b6d6dad5ca4371a9",   "Sensor": "MQ2",   "Raw_Data": 177,   "PPM": 420,   "date": "2021-06-16T15:08:25.643+00:00" }</pre> |

Figura 68: Información de la base de datos