El presente informe detalla el desarrollo de un dispositivo IoT diseñado para medir la concentración de CO2 en el aire, junto con la temperatura y humedad de un ambiente.

El sistema está construido en torno a un microcontrolador ESP32-S, que utiliza un sensor MQ135 para la medición de gases y un sensor AHT25 para obtener lecturas precisas de temperatura y humedad.

Con base en las lecturas de CO2, el dispositivo controla un extractor mediante un relé para mantener la calidad del aire dentro de niveles aceptables.

En los últimos años, la calidad del aire en interiores ha cobrado relevancia debido a su impacto directo en la salud humana. El dióxido de carbono (CO2) es un gas que puede alcanzar concentraciones peligrosas en ambientes cerrados debido a la respiración humana, procesos industriales o combustión. La Organización Mundial de la Salud

(OMS) recomienda mantener las concentraciones de CO2 por debajo de los 1000 ppm (partes por millón) para evitar efectos negativos en la salud, como fatiga, dolor de cabeza y disminución de la productividad. Concentraciones superiores a 5000 ppm pueden ser peligrosas, causando síntomas como dificultad para respirar y deterioro cognitivo. Este proyecto tiene como objetivo desarrollar un dispositivo IoT que monitoree los niveles de CO2, temperatura y humedad, y que active automáticamente un extractor para mantener la calidad del aire dentro de parámetros seguros.

Objetivos

Monitorear con precisión la concentración de CO2 en tiempo real usando el sensor MQ135, calibrado para lecturas más exactas en función de la temperatura y la humedad.

Controlar automáticamente un extractor de aire mediante un relé, activándose cuando la concentración de CO2 supere los 1000 ppm.

Comunicar la información obtenida a través de un bróker MQTT, usando Mosquitto como bróker local y poder visualizar la información en un dispositivo móvil dentro de la red local, mediante la creación de un dashboard con Node-red.

Selección de Sensores

MQ135: El sensor MQ135 es adecuado para medir gases como CO2, amoníaco, benceno y otros compuestos orgánicos volátiles. Su respuesta varía en función de la

concentración de diferentes gases, lo que requiere una calibración específica para detectar CO2 con precisión. Esta calibración debe considerar las lecturas de temperatura y humedad, ya que el sensor es sensible a estos factores (fig. 1).

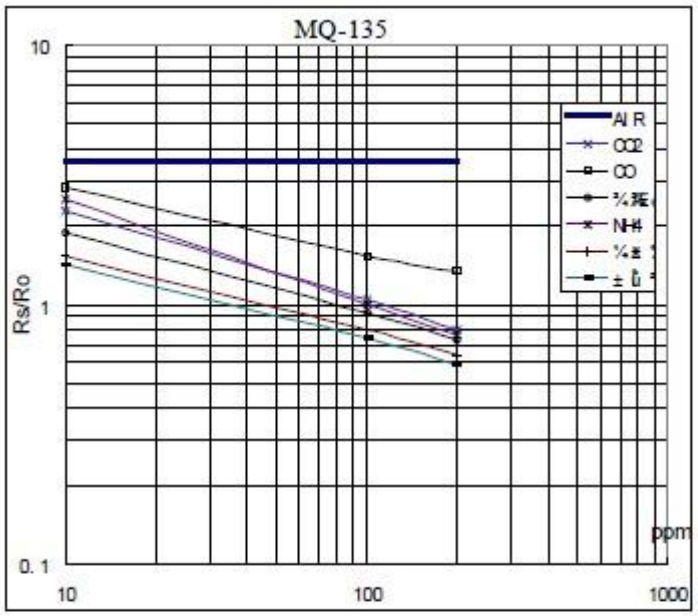


Fig.3 is shows the typical sensitivity characteristics of the MQ-135 for several gases.
in their: Temp: 20℃
Humidity: 65%
O_2 concentration 21%
RL=20kΩ
Ro: sensor resistance at 100ppm of NH_3 in the clean air.
Rs:sensor resistance at various concentrations of gases.

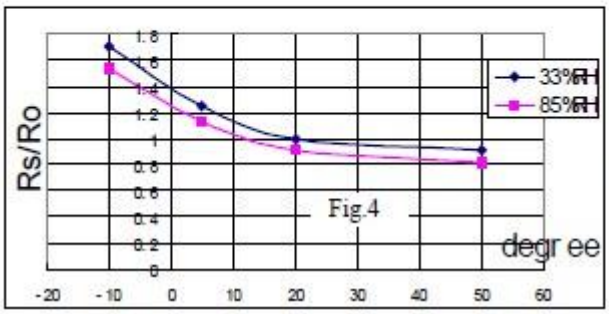


Fig.4 is shows the typical dependence of the MQ-135 on temperature and humidity.
Ro: sensor resistance at 100ppm of NH_3 in air at 33%RH and 20 degree.
Rs: sensor resistance at 100ppm of NH_3 at different temperatures and humidities.

Fig. 1 Respuesta del sensor MQ135 a las concentraciones de distintos gases y su variación con

respecto a la temperatura.

AHT25: El sensor AHT25, es un sensor digital basado en el protocolo I2C, que proporciona mediciones precisas de temperatura y humedad, lo que es esencial para la calibración del MQ135 y para compensar la variabilidad de sus lecturas.

Microcontrolador y Actuador

- **ESP32-S**: Este microcontrolador fue elegido por su conectividad y bajo consumo de energía, facilitando la integración del dispositivo con plataformas en la nube.

Relé y Extractor: Se ha incorporado un relé para controlar un extractor que se activará cuando los niveles de CO2 superen un umbral establecido.

Control del Relé basándose en los Niveles de CO2

Para activar el extractor, el dispositivo debe comparar la lectura de CO2 con un umbral establecido. Si la concentración de CO2 excede ese valor, el sistema activa el relé.

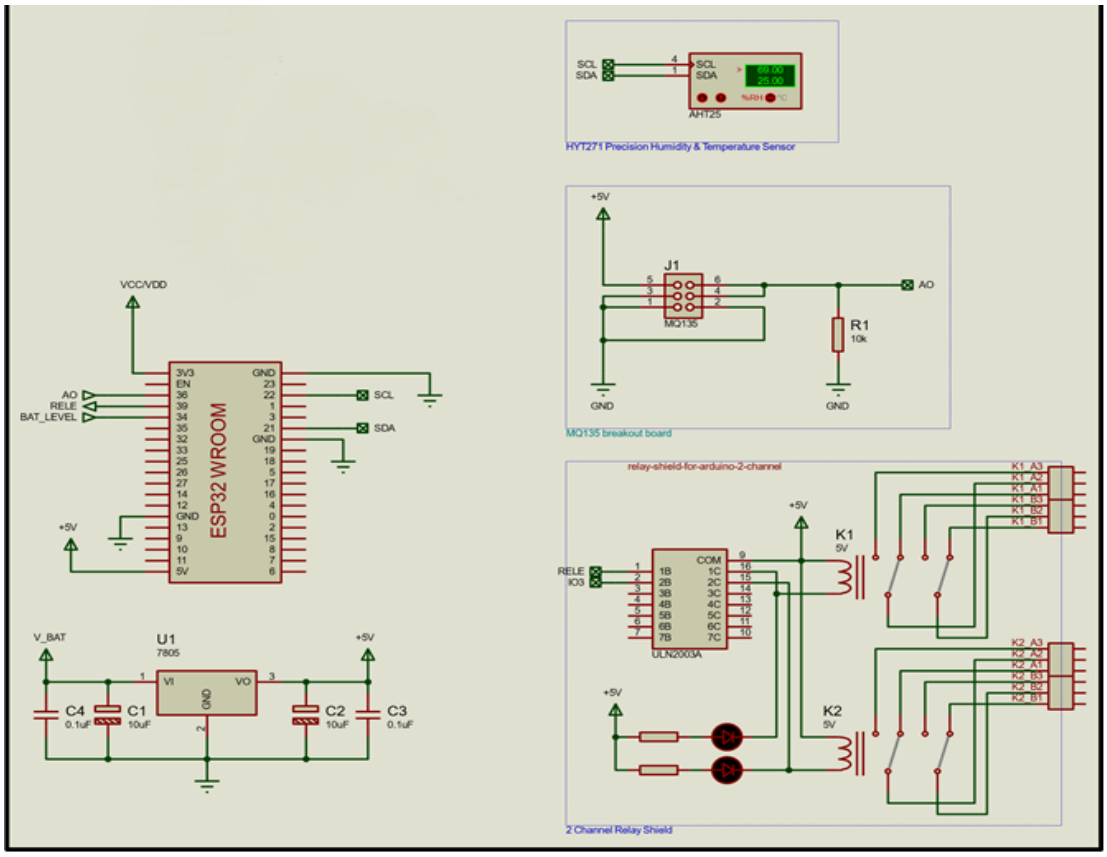
Condición de Activación:

$$\text{si ppm} > \text{umbral} \rightarrow \text{activar relé}$$

En este caso, se recomienda un umbral de 1000 ppm para activar el extractor, ya que este valor es generalmente aceptado como límite superior de calidad de aire en ambientes cerrados según la OMS.

Diseño del circuito

A continuación se muestra el esquema eléctrico del prototipo según lo establecido en los párrafos anteriores.



Comunicación con dashboard

¿Cómo funciona?

1. El **ESP32** lee los valores del sensor de CO_2 (MQ135), temperatura y humedad (AHT25) y el nivel de batería (ADC).
2. Publica todos esos datos por el protocolo **MQTT** hacia un **broker local Mosquitto**.
3. El **broker Mosquitto** es el "repartidor de mensajes". Recibe publicaciones del ESP32 y las reenvía a todos los que estén suscritos.
4. **Node-RED** se conecta como cliente MQTT, escucha los mensajes, los muestra en un **dashboard** web y ejecuta lógica para controlar un **extractor de aire** en caso de concentración elevada de CO_2.
5. El dashboard se puede ver desde cualquier dispositivo dentro de la red local (PC, celular, tablet).

¿Qué es MQTT?

MQTT (Message Queuing Telemetry Transport) es un protocolo de comunicación simple, liviano y orientado a mensajes. Ideal para proyectos IoT.

- Funciona bajo el modelo **publicador/suscriptor**:
  - Un dispositivo **publica** un dato en un *topic* (tema o ruta).
  - Otro dispositivo (o sistema) **se suscribe** a ese *topic* y recibe los datos.
- Los mensajes pasan por un **broker MQTT** (en este caso, Mosquitto en la PC).

Tabla de Topics utilizados

Topic MQTT	Dirección	Función	Ejemplo
`ambiente/co2`	Publicación	Valor en ppm del sensor MQ135	`865.2`
`ambiente/temperatura`	Publicación	Valor de temperatura en °C	`23.6`
`ambiente/humedad`	Publicación	Porcentaje de humedad relativa	`52.4`
`ambiente/extractor/control`	Suscripción	Recibe órdenes para encender/apagar extractor	`ON`, `OFF`
`ambiente/extractor/estado`	Publicación	Informa si el extractor está activo o no	`ON`, `OFF`

Dashboard en Node-RED

Node-RED es una herramienta visual basada en flujos. Permite conectar nodos lógicos como sensores, condiciones, actuadores y visualizadores.

En este proyecto, el dashboard fue creado usando el módulo `node-red-dashboard`. Se diseñó un panel intuitivo para visualizar todas las variables relevantes y controlar el sistema.

Componentes visuales del dashboard:

- `ui_gauge`: Muestra la concentración de CO_2, temperatura y humedad.
- `ui_text`: Muestra el nivel de batería.
- `ui_text`: Muestra el estado del extractor.
- `function`: Evalúa si el CO_2 supera los 1000 ppm y activa el extractor automáticamente.
- `mqtt out`: Envía el comando ON/OFF al topic del extractor.

Detalles importantes del JSON del dashboard:

- Los nodos `mqtt in` están suscritos exactamente a los topics definidos arriba.
- La lógica de control automático del extractor está en un nodo `function` con este contenido:

```
let co2 = parseFloat(msg.payload);
if (co2 > 1000) {
    return { payload: "ON" };
} else {
    return { payload: "OFF" };
}
```

- El dashboard está agrupado dentro de una `ui_group` llamada "Monitoreo", dentro del tab "Panel IoT".
- Todos los mensajes pasan por MQTT: Node-RED **no se comunica directamente con el ESP32**, sino que **lee y escribe a través del broker**.

Acceso al dashboard desde un celular

Node-RED crea un dashboard web accesible en la dirección:

```
http://<IP_DE_LA_PC>:1880/ui
```

Si la PC tiene la IP `192.168.0.101`, entonces desde el celular (conectado a la misma red Wi-Fi) se puede ingresar a:

```
http://192.168.0.101:1880/ui
```

Esto permite usar el sistema en tablets, celulares u otras computadoras sin necesidad de instalar nada más.

Explicación del código modular

El código del ESP32 se dividió en clases para facilitar el mantenimiento y la reutilización:

Archivo	Función
`main.cpp`	Configura sensores, red, broker MQTT, y ejecuta el loop.
`SensorDataMQTT.cpp`	Administra conexión WiFi y MQTT. Publica datos y responde a órdenes.
`Extractor.cpp`	Encapsula la lógica del pin del relé del extractor.
`MQ135Sensor.cpp`	Lectura de concentración de CO_2 con filtrado.
`AHT25Sensor.cpp`	Lectura de temperatura y humedad.

Esta estructura separa responsabilidades y permite que cada módulo se modifique de forma aislada.

A continuación se desglosa las partes importantes del código.

1. **Conexión al broker MQTT**

La conexión se realiza mediante la biblioteca `PubSubClient`. En el constructor de la clase `SensorDataMQTT`, se establece la IP del broker y el puerto:

```
client.setServer(broker, port);
client.setCallback(callbackWrapper);
```

La función `conectarMQTT()` intenta establecer conexión:

```
while (!client.connected()) {
    client.connect("ESP32_Control_CO2");
    delay(1000);
}
```

Aquí `ESP32_Control_CO2` es el `clientID`. Puede ser cualquier identificador único.

Además, se suscribe al topic que controla el extractor:

```
client.subscribe("ambiente/extractor/control");
```

2. Publicación de datos (sensor → MQTT)

Cada 60 segundos, el ESP32 publica las lecturas del CO_2, temperatura, humedad y batería utilizando la función `publish()`:

```
char buffer[16];
dtostrf(co2, 1, 2, buffer);
client.publish("ambiente/co2", buffer);

dtostrf(temperatura, 1, 2, buffer);
client.publish("ambiente/temperatura", buffer);

// ... lo mismo para humedad
```

- Se utiliza `dtostrf()` para convertir los valores float a string.
- Cada dato se publica en un **topic distinto** según el tipo de sensor.

3. Recepción de comandos (MQTT → ESP32)

Cuando Node-RED publica una orden en el topic `ambiente/extractor/control`, el ESP32 la recibe mediante el `callback`.

La librería llama a:

```
client.setCallback(callbackWrapper);
```

Esta redirige a `procesarMensaje()`:

```
if (String(topic) == "ambiente/extractor/control") {
    if (mensaje == "ON") {
        extractor.encender();
        client.publish("ambiente/extractor/estado", "ON");
    } else {
        extractor.apagar();
        client.publish("ambiente/extractor/estado", "OFF");
    }
}
```

Esto permite controlar el extractor **de forma remota y automática**, y además enviar una confirmación de su estado actual al topic `ambiente/extractor/estado`.

Beneficios de la estructura modular

- Toda la lógica de conexión y comunicación MQTT está encapsulada en la clase SensorDataMQTT.
- Esto mejora la organización, facilita el mantenimiento del código y permite escalar el sistema fácilmente.
- Separar la lógica de red del resto del programa permite reutilizar esta clase en otros proyectos IoT sin mayores cambios.

El Dashboard

El Dashboard fue creado con **Node-RED** usando el complemento node-red-dashboard, que permite generar interfaces web fácilmente conectadas al flujo de datos.

A continuación, se explican los elementos principales del JSON que compone el Dashboard.

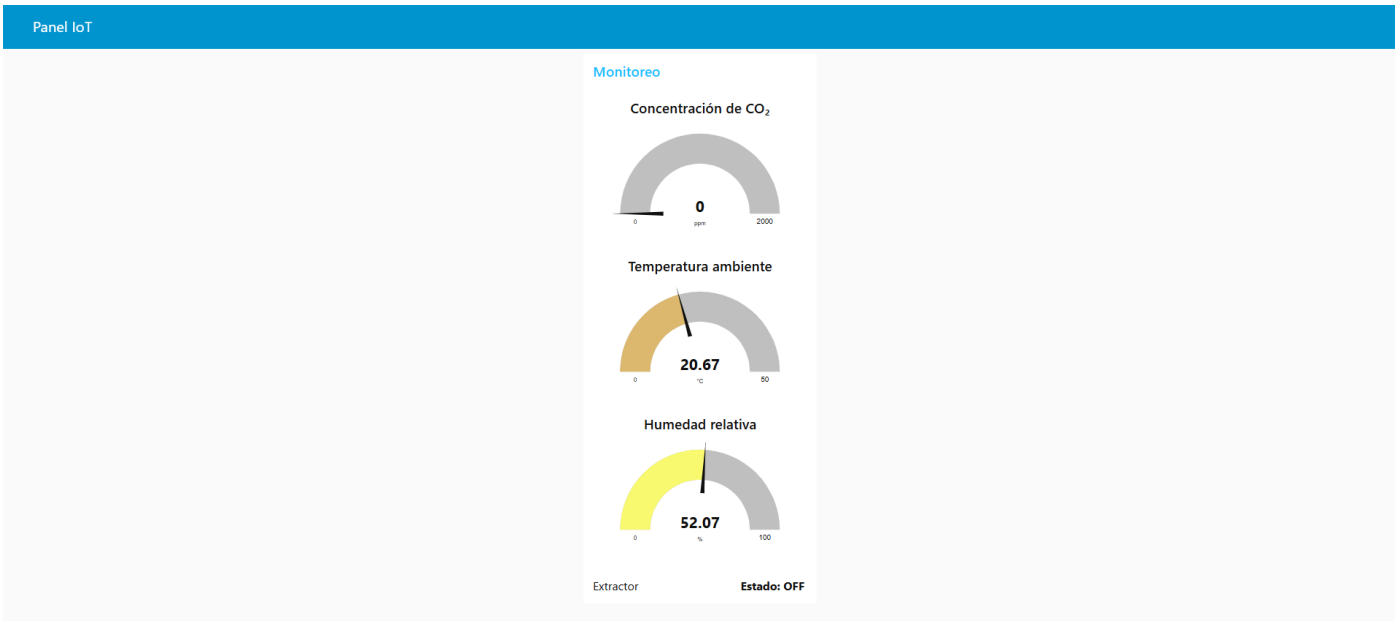
1. Definición del Tab y Grupo

```
{
  "id": "tab_principal",
  "type": "ui_tab",
  "name": "Panel IoT",
  "icon": "dashboard"
}
```

- **ui_tab** define una pestaña principal llamada *"Panel IoT"*.
- Todos los elementos visuales estarán organizados dentro de esta pestaña.

```
{
  "id": "grupo_panel",
  "type": "ui_group",
  "name": "Monitoreo",
  "tab": "tab_principal"
}
```

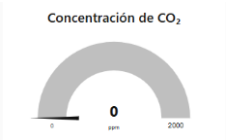
- **ui_group** crea un grupo de elementos visuales dentro del tab.
- Este grupo se llama *"Monitoreo"* y contiene todos los indicadores del sistema.



2. Indicadores de datos (Gauges y texto)

- ✓ **CO_2 (ppm)**

```
{
  "type": "ui_gauge",
  "name": "CO2 (ppm)",
  "topic": "ambiente/co2",
  "group": "grupo_panel",
  "title": "Concentración de CO2",
  "label": "ppm",
  "min": 0,
  "max": 2000
}
```



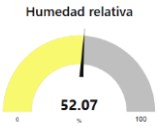
- Gauge circular que muestra el valor en tiempo real del sensor MQ135.
- Se suscribe al topic ambiente/co2.

- ✓ **Temperatura**

```
{
  "type": "ui_gauge",
  "name": "Temperatura",
  "topic": "ambiente/temperatura",
  "title": "Temperatura ambiente",
  "label": "°C",
  "min": 0,
  "max": 50
}
```

- ✓ **Humedad**

```
{
  "type": "ui_gauge",
  "name": "Humedad",
  "topic": "ambiente/humedad",
  "title": "Humedad relativa",
  "label": "%"
}
```



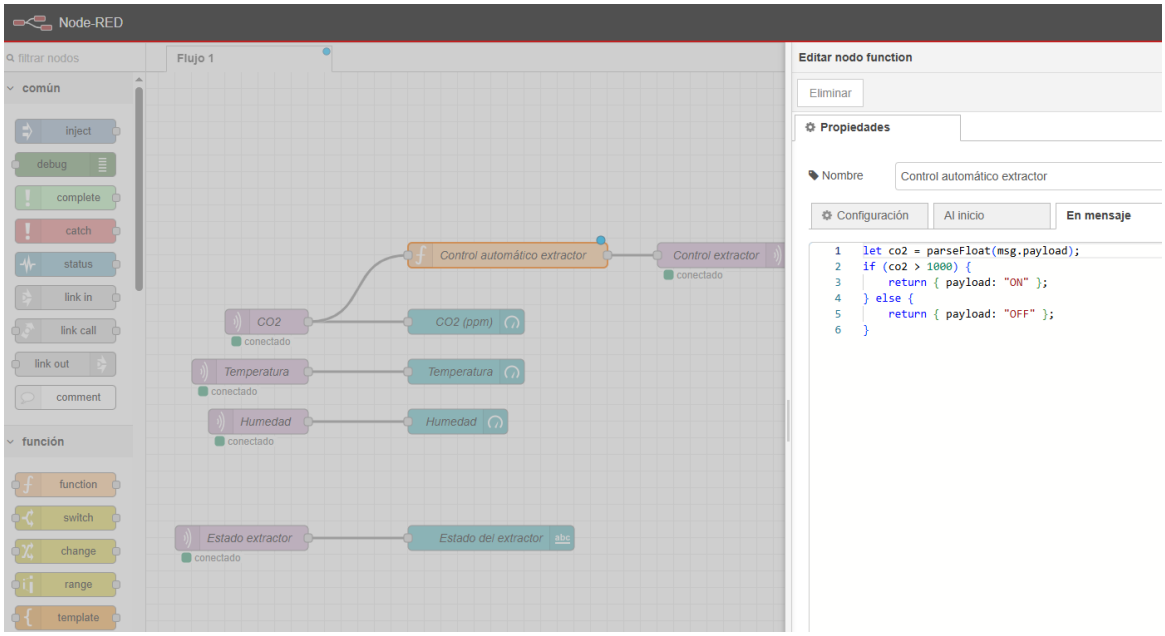
3. Control automático del extractor

```
{
  "type": "function",
  "name": "Control automático extractor",
  "func": "let co2 = parseFloat(msg.payload);\nif (co2 > 1000) {\n    return { payload: \"ON\" };\n} else {\n    return { payload: \"OFF\" };\n}"
}
```

- Lógica embebida en un nodo function.
- Si el valor de CO_2 supera los **1000 ppm**, el nodo publica "ON".

```
{
  "type": "mqtt out",
  "topic": "ambiente/extractor/control"
}
```

- El resultado del nodo anterior se envía al ESP32 mediante MQTT.
- Esto enciende o apaga el extractor automáticamente.



4. Estado del extractor

```
{
  "type": "mqtt in",
  "topic": "ambiente/extractor/estado"
}
```

- El ESP32 informa si el extractor está encendido o apagado.

```
{
  "type": "ui_text",
  "label": "Extractor",
```

```
  "format": "Estado: {{msg.payload}}"
}
```

- El estado se muestra como texto simple en el dashboard.

5. Acceso desde celular

El Dashboard está disponible en:

`http://<IP_DEL_PC>:1880/ui`

Ejemplo:

`http://192.168.0.101:1880/ui`

- Solo es necesario que el celular esté conectado a la **misma red Wi-Fi**.
- No hace falta instalar nada: se accede desde el navegador.