

# Sistema IoT para Monitoreo de Calidad de Aire y Gas Metano en una Vivienda de Madera utilizando ESP32, MQ-4, MQ-135 y Google Colab

Miguel Ángel Ramírez Beltrán  
Ingeniería Electrónica  
Universidad Distrital Francisco José de Caldas  
Bogotá, Colombia  
Código: 20241005112

Miguel Stiven Rodríguez Hortua  
Ingeniería Electrónica  
Universidad Distrital Francisco José de Caldas  
Bogotá, Colombia  
Código: 20241005173

**Resumen**—Este artículo presenta el diseño e implementación de un sistema IoT basado en ESP32 para el monitoreo de calidad de aire y detección de gas metano dentro de una vivienda de madera a escala. Para este propósito se utilizan los sensores MQ-4 (detección de metano) y MQ-135 (calidad de aire). Los datos son enviados a Google Sheets mediante peticiones HTTP para posteriormente ser procesados y graficados desde Google Colab. Asimismo, el sistema se integra a la plataforma Ubidots para visualización en tiempo real y control remoto de un extractor de aire encargado de ventilar el ambiente cuando se detectan niveles inseguros. Se presenta la arquitectura del sistema, el análisis de pruebas realizadas y el funcionamiento detallado del código.

**Index Terms**—IoT, ESP32, MQ-4, MQ-135, Google Colab, Ubidots, gas metano, calidad de aire, ventilación automática.

## I. INTRODUCCIÓN

El monitoreo de la calidad del aire en espacios cerrados es fundamental para garantizar condiciones saludables, especialmente en viviendas de madera donde la acumulación de gases puede representar un riesgo. Sensores como el MQ-4 y MQ-135 son ampliamente utilizados en aplicaciones de bajo costo para la detección de gases combustibles y contaminantes. La incorporación de plataformas IoT permite visualizar datos en tiempo real y actuar automáticamente ante condiciones anómalas.

Este proyecto desarrolla una solución embebida que integra sensores analógicos, una ESP32, motores controlados mediante L298N y plataformas como Google Colab y Ubidots para análisis y control. La vivienda a escala sirve como entorno experimental para validar el comportamiento del sistema.

## II. MARCO TEÓRICO

### II-A. Sensor MQ-4

El sensor MQ-4 es un detector especializado en gas metano ( $\text{CH}_4$ ), diseñado para medir concentraciones típicas en aplicaciones domésticas e industriales. Su respuesta es rápida y tiene buena sensibilidad en rangos entre 200 y 10 000 ppm.

### II-B. Sensor MQ-135

El MQ-135 permite medir la calidad general del aire, reaccionando a gases como amoníaco, óxidos de nitrógeno,

alcohol, benceno y  $\text{CO}_2$ . Su salida es analógica y responde notablemente ante cambios generados por el aliento humano, como se evidenció en pruebas experimentales.

### II-C. ESP32 e IoT

La ESP32 cuenta con WiFi integrado, lo que facilita el envío de datos a servicios web. Se emplean peticiones HTTP y el servicio Ubidots para telemetría y control.

## III. METODOLOGÍA DE IMPLEMENTACIÓN

### III-A. Arquitectura General

El sistema se compone de:

- Sensores MQ-4 y MQ-135 conectados a las entradas ADC de la ESP32.
- Módulo controlador de motor (L298N) para el extractor.
- Pantalla LCD 16x2 para visualización local.
- Conexión WiFi para envío de datos a Google Sheets.
- Análisis remoto en Google Colab.
- Control IoT con Ubidots.

### III-B. Flujo de operación

1. La ESP32 toma muestras de los sensores.
2. Se envían valores a Google Sheets mediante Apps Script.
3. Google Colab accede al archivo para graficar las mediciones.
4. Ubidots recibe estados y permite controlar el extractor remotamente.
5. El ventilador se activa si la calidad de aire es deficiente.

## IV. RESULTADOS EXPERIMENTALES

Durante las pruebas se observó que:

- El MQ-4 responde de forma precisa únicamente ante metano.
- El MQ-135 registra variaciones sensibles al aliento humano.
- El extractor responde correctamente al comando IoT.
- La visualización en Google Colab facilita el análisis de tendencias.

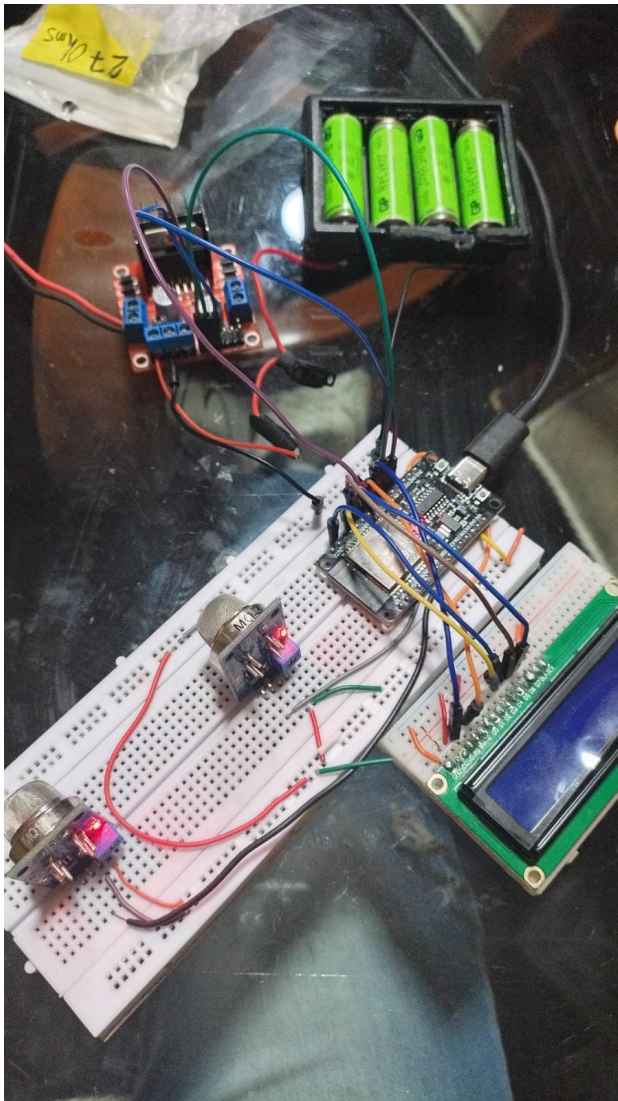


Figura 1. montaje general del sistema (espacio para imagen).

## V. ESPACIOS PARA FIGURAS

### VI. CÓDIGO FUENTE

#### VI-A. A. Código principal ESP32 (C++)

```

1 #include <WiFi.h>
2 #include <HTTPClient.h>
3 #include <UbiConstants.h>
4 #include <UbidotsEsp32Mqtt.h>
5 #include <UbiTypes.h>
6 #include <LiquidCrystal.h>
7
8 const char* ssid = "Miguel12";
9 const char* password = "123456789";
10
11 String baseURLExtractor = "https://script.google.com
    /macros/s/AKfycbz7UX4f768
12 OjtXpDevOBtOjlcZKhXoLwR3E_ILBuh19D-
    Aj8oN43uaAaneOk3xdQLPf/exec";
13 int pinSenExtractor = 34;
14 int NExtractor = 10;
15

```

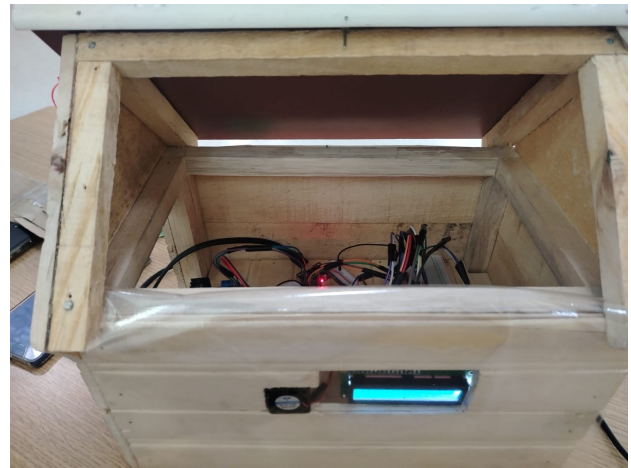


Figura 2. Montaje físico de la mini vivienda (espacio para imagen).

```

16 String baseURLSensor2 = "https://script.google.com/
    macros/s/
17 AKfycbzOLL7Uk4qt0SD0HwFCpHZuBDWLNxn9WXeH
    6h3yiQBv2EZSazdrTWjBhJc_2WnOyZ9Nfw/exec";
18 int pinSen2 = 32;
19 int N2 = 10;
20
21 #define UBIDOTS_TOKEN "BBUS-
    GfOVgA8n2YOMayNTdq7ap6lY4bUTSW"
22 #define DEVICE_LABEL "extractoresp32"
23 Ubidots ubidots(UBIDOTS_TOKEN);
24
25 const int in1 = 2;
26 const int in2 = 4;
27 const int en = 5;
28
29 LiquidCrystal lcd(16, 17, 21, 22, 18, 19);
30
31 bool extractorDisponibleMostrado = false;
32 int estadoExtractor = 0;
33
34 void mostrarExtractorDisponible() {
35     lcd.clear();
36     lcd.setCursor(0,0);
37     lcd.print("EXTRACTOR");
38     lcd.setCursor(0,1);
39     lcd.print("DISPONIBLE");
40     extractorDisponibleMostrado = true;
41 }
42
43 // ...
44
45 // ...
46
47 void loop() {
48     if (!ubidots.connected()) {
49         ubidots.reconnect();
50         ubidots.subscribeLastValue(DEVICE_LABEL, "
51             extractor");
52     }
53     ubidots.loop();
54 }
55

```

#### VI-B. C. Librerías utilizadas (espacio reservado)

- **WiFi.h:** Utilizada para conectar la red wifi con la ESP32
- **HTTPClient.h:** Se utilizó para que los sensores envíen y reciban datos a través del internet, se usa para que la ESP32 se conecte al url y pueda enviar los datos.

- UbidotsEsp32Mqtt.h:Para conectar con la Plataforma Utilizada para el Iot
- LiquidCrystal.h:Para controlar la pantalla LCD 16x2.

#### VI-C. Codigo Realizado en Google colab para el Sensor MQ-4

```

1 //conectar drive
2 from google.colab import drive
3 drive.mount('/content/drive')
4
5
6 //conectar con google sheets(hoja de calculo)
7
8 from google.colab import drive
9 import pandas as pd
10
11 drive.mount('/content/drive')
12
13 sheet_url = "https://docs.google.com/spreadsheets/d
14 /1LMC9NHaoLZff9QaMXOmKPbbBUDa3AyKfKMoDrPqNLU0/
15 export?format=csv&gid=0"
16 df = pd.read_csv(sheet_url)
17
18 df.to_csv("/content/drive/MyDrive/Proyecto_Final/MQ4
19 .csv", index=False)4
20
21 //Graficar datos contenidos en google sheet
22 import pandas as pd
23 import matplotlib.pyplot as plt
24
25 url = "https://docs.google.com/spreadsheets/d/1
26 LMC9NHaoLZff9QaMXOmKPbbBUDa3AyKfKMoDrPqNLU0/
27 export?format=csv"
28
29 df = pd.read_csv(url)
30
31 plt.plot(df["tiempo"], df["valor"])
32 plt.xlabel("Tiempo_(ms)")
33 plt.ylabel("Valor_ADC")
34 plt.title("Datos_del_sensor_MQ4")
35 plt.grid()
36 plt.show()
37
38 //calcular el promedio
39 url_mq4 = "https://docs.google.com/spreadsheets/d/1
40 LMC9NHaoLZff9QaMXOmKPbbBUDa3AyKfKMoDrPqNLU0/gviz
41 /tq?tqx=out:csv&sheet=Hoja1"
42 df_mq4 = pd.read_csv(url_mq4)
43 promedio_mq4 = df_mq4["valor"].mean()
44 print("Promedio_MQ4:", promedio_mq4)
45
46 umbral_peligro = 1100
47 if promedio_mq4 > umbral_peligro:
48     print("\n_Nivel_Elevado_de_gas_Metano_")
49 else:
50     print("\n_Niveles_seguros")

```

#### VI-D. Codigo Realizado en google colab para el Sensor MQ-135

```

1 /Conectar el google colab con el google sheets para
2 el mq-135
3 import pandas as pd
4 from google.colab import drive
5 drive.mount('/content/drive')
6
7 sheet_id = "1DP9AI7-
8 RHRyJyJDMFLgP9hNI_3cH0HM2BZKplbgQH0"

```

```

sheet_name = "Hoja2"

sheet_url = f"https://docs.google.com/spreadsheets/d
/{sheet_id}/gviz/tq?tqx=out:csv&sheet={
sheet_name}"

df = pd.read_csv(sheet_url)

df.to_csv("/content/drive/MyDrive/Proyecto_Final/
MQ135.csv", index=False)

//Para que lea los datos de google sheet y los
grafique

import pandas as pd
import matplotlib.pyplot as plt

url = "https://docs.google.com/spreadsheets/d/1
DP9AI7-RHRyJyJDMFLgP9hNI_3cH0HM2BZKplbgQH0/gviz
/tq?tqx=out:csv&sheet=Hoja2"

df = pd.read_csv(url)

plt.plot(df["tiempo"], df["valor"])
plt.xlabel("Tiempo_(ms)")
plt.ylabel("Valor_ADC")
plt.title("Datos_del_sensor_MQ135")
plt.grid()
plt.show()

//Para calcular el promedio

url_mq135 = "https://docs.google.com/spreadsheets/d
/1DP9AI7-RHRyJyJDMFLgP9hNI_3cH0HM2BZKplbgQH0/
gviz/tq?tqx=out:csv&sheet=Hoja2"

df_mq135 = pd.read_csv(url_mq135)
promedio_mq135 = df_mq135["valor"].mean()
print("Promedio_MQ135:", promedio_mq135)

//Para que lea el promedio e indique el nivel de gas
umbral_peligro2 = 1100
if promedio_mq135 < umbral_peligro2:
    print("\n_El_aire_se_encuentra_un_poco_
contaminado_")
else:
    print("\n_El_aire_se_encuentra_seguro_para_
habitar")

//Para normalizar el promedio de los 2 sensores y
asi calcular un promedio general
MQ4_normal=promedio_mq4/umbral_peligro
print("Promedio_MQ4_normalizado:", MQ4_normal)

MQ135_normal=umbral_peligro2/promedio_mq135
print("Promedio_MQ135_normalizado:", MQ135_normal)

PromedioGeneral=(MQ4_normal+MQ135_normal)/2
print("Promedio_General:", PromedioGeneral)

if PromedioGeneral >1:
    print("\n_Aire_Contaminado,_iniciar_proceso_de_
extracci_n_")
else:
    print("\n_Aire_seguro,_puede_seguir_con_
tranquilidad")

```

El código principal ejecutado en la ESP32 se encarga de leer continuamente los valores analógicos provenientes de los sensores MQ-4 y MQ-135. Cada dato medido es enviado hacia Google Sheets mediante solicitudes HTTP GET dirigidas a un Web App generado en Google Apps Script, el cual recibe los parámetros y los almacena directamente en las hojas de cálculo

correspondientes. Este proceso permite mantener un registro continuo y estructurado de las mediciones.

Google Colab se utiliza como herramienta de análisis y visualización. Para ello, primero se establece una conexión con Google Drive mediante la instrucción `drive.mount()`, lo que permite acceder al archivo generado en Sheets. Posteriormente, Colab descarga los datos utilizando la URL pública de la hoja, empleando el parámetro `export?format=csv` para convertirla automáticamente en un archivo CSV procesable. Las librerías *pandas* y *matplotlib* permiten cargar el conjunto de datos, manipularlo y generar gráficas que muestran la evolución temporal de las mediciones.

El flujo típico consiste en leer la hoja de cálculo, convertirla en un *DataFrame* y trazar la señal del sensor (por ejemplo, valor ADC frente al tiempo). Este mecanismo posibilita un análisis visual inmediato del comportamiento del MQ-4 y MQ-135, permitiendo identificar tendencias, variaciones rápidas o comportamientos anómalos durante las pruebas.

Además, la plataforma Ubidots se integra mediante MQTT para habilitar el control remoto del extractor de aire. La ESP32 recibe comandos en tiempo real provenientes de la nube, y a través del controlador L298N se activa o desactiva el motor según el estado enviado por el usuario. La pantalla LCD incorporada muestra continuamente el estado del sistema, los valores detectados y las acciones ejecutadas, brindando retroalimentación local y clara.

En conjunto, la comunicación entre ESP32, Google Sheets, Google Colab y Ubidots conforma un sistema IoT completo: captura de datos, almacenamiento en la nube, análisis automatizado y control inteligente del actuador.

## VII. CONCLUSIONES

El sistema desarrollado demostró la capacidad de integrar sensores de bajo costo con una plataforma IoT robusta y accesible. La ESP32 permitió una gestión eficiente de datos, mientras que Google Colab facilitó el análisis y procesamiento. La automatización del extractor garantiza una respuesta inmediata ante condiciones adversas, lo que hace del sistema una solución viable para entornos domésticos.

## VIII. REFERENCIAS

### REFERENCIAS

- [1] M. Patel, "MQ Sensors Based Air Quality Monitoring System," IEEE Xplore, 2021.
- [2] J. Khan, "IoT-Based Gas Leakage Detection using ESP32," IEEE Xplore, 2020.
- [3] Scimago Journal & Country Rank, "Environmental Monitoring and Assessment," 2023.
- [4] A. Sharma, "Integration of Google Cloud Services for IoT Monitoring," IEEE Xplore, 2022.
- [5] L. Torres, "Low-Cost Air Quality Monitoring using MQ Sensors," IEEE, 2019.