

MQ135 Sensor Overview

MQ135 Sensor tell about that

The **MQ135** is an air quality sensor used to detect a wide range of gases, including **ammonia** (NH_3), **sulfur dioxide** (SO_2), **benzene**, **carbon dioxide** (CO_2), and **smoke**. It is commonly used in air quality monitoring systems, gas leak detection, and environmental applications.

Key Features:

- Detects **various gases** (NH_3 , NO_x , alcohol, benzene, smoke, CO_2 , etc.)
- **Analog & digital output**
- **Low power consumption**
- **Preheat time:** Requires a warm-up period (~24 hours for accurate readings)

Working Principle:

The sensor contains a **SnO_2 (tin dioxide) sensing layer**, which changes resistance when exposed to gases. This resistance change is converted into an electrical signal, which can be read by a microcontroller (e.g., Arduino, ESP32, Raspberry Pi).

Connections with Arduino:

- **VCC** → 5V
- **GND** → Ground
- **AO (Analog Output)** → Analog pin of the microcontroller
- **DO (Digital Output)** → Digital pin (for threshold-based gas detection)

Applications:

- Indoor & outdoor air quality monitoring
- Gas leak detection in industries
- Smart home automation (detecting harmful gases)
- Environmental pollution monitoring

Would you like a sample Arduino code for using the MQ135?

twll about carbon monoxide sensor(SEN0132)

The **SEN0132** is a **carbon monoxide (CO) gas sensor** module based on the **MQ7** gas sensor. It is used to detect **carbon monoxide (CO) concentration** in the air and is widely used in gas leak detection systems, safety alarms, and industrial applications.

Key Features:

- Detects **carbon monoxide (CO) gas** in the range of **10–1000 ppm**
- Uses **MQ7 sensor**, which requires a heating cycle for accurate readings
- **Analog output (AO)** and **digital output (DO)**
- Operates at **5V DC**
- **High sensitivity & fast response**

Working Principle:

The MQ7 sensor inside the SEN0132 module works based on a **SnO₂ (tin dioxide) sensing layer**, which changes resistance in response to CO gas. It operates in a **heating cycle**, alternating between **high temperature (5V)** for cleaning and **low temperature (1.4V)** for detection.

Pinout:

- **VCC → 5V (Power)**
- **GND → Ground**
- **A0 (Analog Output) → Connect to an analog pin** of a microcontroller
- **D0 (Digital Output) → Connect to a digital pin** for threshold-based alerts

Applications:

- Carbon monoxide leak detection
- Home and industrial safety systems
- Air quality monitoring
- Smart home automation

Would you like an **Arduino code example** for the SEN0132 sensor?

i want to do air quality monitor project i have raspberry pi and tell me what sensors can we use for my project and

For an **air quality monitoring** project using a **Raspberry Pi**, you can use affordable sensors to measure various air pollutants. Here are some **low-cost** sensor options:

1. MQ135 (Air Quality Sensor)

- **Detects:** CO₂, NH₃, NO_x, alcohol, benzene, smoke
- **Cost:** ₹250–₹500
- **Interface:** Analog output (requires ADC for Raspberry Pi)
- **Best for:** General air quality monitoring

2. MQ7 (Carbon Monoxide Sensor)

- **Detects:** Carbon Monoxide (CO)
- **Cost:** ₹200–₹400
- **Interface:** Analog output (needs ADC for Raspberry Pi)
- **Best for:** CO leak detection

3. MQ2 (Gas & Smoke Sensor)

- **Detects:** Smoke, methane, butane, propane
- **Cost:** ₹150–₹300
- **Interface:** Analog output (requires ADC)
- **Best for:** Smoke and LPG leakage detection

4. PMS5003 / PMS7003 (Particulate Matter Sensor)

- **Detects:** PM2.5, PM10 (dust particles)
- **Cost:** ₹1,200–₹2,000
- **Interface:** UART (directly compatible with Raspberry Pi)
- **Best for:** Dust and pollution monitoring

5. DHT22 / DHT11 (Temperature & Humidity Sensor)

- **Measures:** Temperature & humidity
- **Cost:** ₹200–₹500

- **Interface:** Digital (directly compatible with Raspberry Pi)
- **Best for:** Monitoring weather conditions along with air quality

6. BME680 (Air Quality & Weather Sensor)

- **Detects:** VOCs (Volatile Organic Compounds), temperature, humidity, pressure
- **Cost:** ₹1,500–₹2,500
- **Interface:** I2C/SPI (directly compatible with Raspberry Pi)
- **Best for:** Indoor air quality monitoring

How to Connect to Raspberry Pi?

Since MQ series sensors give **analog output**, you'll need an **ADC module** (like MCP3008) to read data with the Raspberry Pi.

Suggested Budget Combination (Low Cost)

- MQ135 (Air Quality) – ₹300
- MQ7 (CO detection) – ₹250
- DHT11 (Temperature & Humidity) – ₹200
- MCP3008 (ADC for MQ sensors) – ₹250
- **Total Approximate Cost:** ₹1,000

Would you like a **circuit diagram** or **Python code** to get sensor data on Raspberry Pi?

now we have raspberry pi, df robot io expansion and some sensors MQ7 (Carbon Monoxide Sensor), soil moisture sensor, temperature sensor, steam sensor by using above sensors tell me how to implement the complete project and i want also visualizations into my mobile.

2/2

To build a **complete air quality monitoring system** using your **Raspberry Pi**, **DF Robot IO Expansion**, and sensors (MQ7, soil moisture, temperature, and steam sensor) with **visualization on mobile**, follow this structured approach:

◆ Project Implementation Plan

1 Hardware Setup

Components Required:

- Raspberry Pi (Model 3/4 recommended)
- DFRobot IO Expansion Shield (for easy sensor connections)
- MQ7 Carbon Monoxide Sensor (Air quality monitoring)
- Soil Moisture Sensor (For additional environmental monitoring)
- Temperature Sensor (e.g., DHT11/DHT22) (Temperature & Humidity data)
- Steam Sensor (Detects vapor in the air)
- MCP3008 ADC Module (Required for MQ7 & soil moisture sensors, as Raspberry Pi lacks an analog input)
- Jumper Wires & Power Supply

2 Wiring & Sensor Connections

◆ DF Robot IO Expansion Shield Pin Connections:

Sensor	Type	Raspberry Pi Pin (via IO Expansion)
MQ7	Analog	Connect via MCP3008 (CH0)
Soil Moisture	Analog	Connect via MCP3008 (CH1)
Temperature (DHT11/DHT22)	Digital	GPIO 4
Steam Sensor	Digital	GPIO 17

◆ MCP3008 (ADC) Wiring for Analog Sensors (MQ7, Soil Moisture)

MCP3008 Pin	Raspberry Pi Pin
VDD	3.3V
VREF	3.3V
AGND	GND
CLK	GPIO 11 (SPI CLK)
DOUT	GPIO 9 (SPI MISO)

MCP3008 Pin	Raspberry Pi Pin
DIN	GPIO 10 (SPI MOSI)
CS/SHDN	GPIO 8 (SPI CE0)

Enable SPI on Raspberry Pi:

Run `sudo raspi-config` → Interface Options → Enable SPI

3 Software Setup

♦ Install Required Libraries

```
bash

sudo apt update && sudo apt install python3-pip
pip install RPi.GPIO spidev adafruit_dht flask requests
```

♦ Read Sensor Data (Python Script)

```
python

import RPi.GPIO as GPIO
import spidev
import Adafruit_DHT
import time
import requests

# Sensor & Pin Configurations
DHT_SENSOR = Adafruit_DHT.DHT11
DHT_PIN = 4
SOIL_MOISTURE_CHANNEL = 1 # MCP3008 Channel for Soil
MQ7_CHANNEL = 0 # MCP3008 Channel for MQ7
STEAM_SENSOR_PIN = 17 # Digital Input

# Initialize SPI (for ADC MCP3008)
spi = spidev.SpiDev()
spi.open(0, 0)
spi.max_speed_hz = 1350000

GPIO.setmode(GPIO.BCM)
```

```

GPIO.setup(STEAM_SENSOR_PIN, GPIO.IN)

# Read Analog Data from MCP3008
def read_adc(channel):
    adc = spi.xfer2([1, (8 + channel) << 4, 0])
    data = ((adc[1] & 3) << 8) + adc[2]
    return data

# Read Sensors
def get_sensor_data():
    temp, humidity = Adafruit_DHT.read(DHT_SENSOR, DHT_PIN)
    mq7_value = read_adc(MQ7_CHANNEL)
    soil_moisture = read_adc(SOIL_MOISTURE_CHANNEL)
    steam_status = GPIO.input(STEAM_SENSOR_PIN)

    data = {
        "temperature": temp,
        "humidity": humidity,
        "co_level": mq7_value,
        "soil_moisture": soil_moisture,
        "steam_detected": steam_status
    }

    print("Data:", data)

# Send data to web server for visualization
requests.post("http://your-server-ip:5000/update", json=data)

return data

# Continuous Monitoring
while True:
    get_sensor_data()
    time.sleep(5) # Read every 5 seconds

```

4 Web Dashboard & Mobile Visualization

To visualize data on **mobile**, we'll use a **Flask web server** on Raspberry Pi, and access it from a mobile browser.

♦ Install Flask (Web Framework)

```
bash

pip install flask
```

♦ Create a Flask Web Server

```
python

from flask import Flask, request, jsonify, render_template

app = Flask(__name__)

# Store last sensor values
sensor_data = {"temperature": 0, "humidity": 0, "co_level": 0, "soil_moisture": 0,
               "steam_detected": 0}

@app.route("/")
def index():
    return render_template("dashboard.html", data=sensor_data)

@app.route("/update", methods=["POST"])
def update_data():
    global sensor_data
    sensor_data = request.json
    return jsonify({"status": "updated"})

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000, debug=True)
```

5 Create a Web Dashboard (HTML & JavaScript)

Create a file named `templates/dashboard.html`

```
html

<!DOCTYPE html>
<html lang="en">
<head>
```



```

<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Air Quality Monitor</title>
<script>
    function fetchData() {
        fetch('/update')
            .then(response => response.json())
            .then(data => {
                document.getElementById("temp").innerText = data.temperature + "
°C";

                document.getElementById("humidity").innerText = data.humidity +
" %";

                document.getElementById("co").innerText = data.co_level;
                document.getElementById("soil").innerText = data.soil_moisture;
                document.getElementById("steam").innerText = data.steam_detected
? "Yes" : "No";
            });
        setInterval(fetchData, 3000);
    }
</script>
</head>
<body>
    <h1>Air Quality Monitor</h1>
    <p><b>Temperature:</b> <span id="temp">Loading...</span></p>
    <p><b>Humidity:</b> <span id="humidity">Loading...</span></p>
    <p><b>CO Level:</b> <span id="co">Loading...</span></p>
    <p><b>Soil Moisture:</b> <span id="soil">Loading...</span></p>
    <p><b>Steam Detected:</b> <span id="steam">Loading...</span></p>
</body>
</html>

```

6 Access Dashboard on Mobile

- Run the Flask server:

```
bash
```

```
python3 your_script.py
```

- **Find Raspberry Pi's IP Address:**

```
bash
```

```
hostname -I
```

- **Access Dashboard from Mobile:**

Open `http://<RaspberryPi_IP>:5000` in your mobile browser.

Summary

1. **Hardware Setup** (Connect sensors via IO expansion + MCP3008 for analog sensors).
2. **Python Script** to read data and send it to a web server.
3. **Flask Web App** to visualize real-time data.
4. **Access Dashboard** on mobile via a browser.

Would you like **additional features**, such as **data logging** or **notifications on mobile**? 

tell me i want to use firebase and also i have LM35 temperature sensor v4 tell according to that

Since you want to use **Firestore** for real-time data storage and have an **LM35 Temperature Sensor V4**, I'll modify the implementation accordingly.

Updated Air Quality Monitoring System (Raspberry Pi + Firebase + LM35)

You will:

- ✓ **Read sensor data** (MQ7, Soil Moisture, LM35, Steam Sensor)
- ✓ **Send data to Firebase** (Realtime Database)
- ✓ **Visualize data on a mobile-friendly web app**

◆ 1. Required Components

- Raspberry Pi (Model 3/4)
- DFRobot IO Expansion (for easy sensor connections)
- MQ7 (Carbon Monoxide Sensor) (Analog input via ADC)
- Soil Moisture Sensor (Analog input via ADC)
- LM35 (Temperature Sensor V4) (Analog input via ADC)
- Steam Sensor (Digital input)
- MCP3008 ADC Module (Required for analog sensors)
- Jumper Wires & Power Supply

◆ 2. Wiring & Pin Configuration

MCP3008 (ADC) Wiring for Analog Sensors (MQ7, Soil Moisture, LM35)

MCP3008 Pin	Raspberry Pi Pin
VDD	3.3V
VREF	3.3V
AGND	GND
CLK	GPIO 11 (SPI CLK)
DOUT	GPIO 9 (SPI MISO)
DIN	GPIO 10 (SPI MOSI)
CS/SHDN	GPIO 8 (SPI CE0)

Sensor Pin Assignments

Sensor	Type	Raspberry Pi Pin (via IO Expansion)
MQ7	Analog	MCP3008 (CH0)
Soil Moisture	Analog	MCP3008 (CH1)
LM35 (Temp Sensor)	Analog	MCP3008 (CH2)
Steam Sensor	Digital	GPIO 17

◆ 3. Firebase Setup

1. Go to [Firebase Console](#)
2. Create a new project
3. Go to **Project Settings** → **Service Accounts** → Generate a private key (JSON file)
4. Install Firebase Python library:

```
bash

pip install firebase-admin spidev RPi.GPIO
```

◆ 4. Python Script (Send Data to Firebase)

```
python

import RPi.GPIO as GPIO
import spidev
import firebase_admin
from firebase_admin import credentials, db
import time

# Firebase Setup
cred = credentials.Certificate("path/to/your/firebase-private-key.json") # Update
this path
firebase_admin.initialize_app(cred, {'databaseURL': "https://your-
project.firebaseio.com/"})
ref = db.reference("/sensor_data")

# SPI Setup for MCP3008 ADC
spi = spidev.SpiDev()
spi.open(0, 0)
spi.max_speed_hz = 1350000

GPIO.setmode(GPIO.BCM)
STEAM_SENSOR_PIN = 17
```

```

GPIO.setup(STEAM_SENSOR_PIN, GPIO.IN)

# Read Analog Data from MCP3008
def read_adc(channel):
    adc = spi.xfer2([1, (8 + channel) << 4, 0])
    data = ((adc[1] & 3) << 8) + adc[2]
    return data

# Read Sensors & Send Data to Firebase
def upload_sensor_data():
    mq7_value = read_adc(0)
    soil_moisture = read_adc(1)
    lm35_raw = read_adc(2)
    temperature = (lm35_raw * 3.3 / 1024) * 100 # LM35 conversion
    steam_status = GPIO.input(STEAM_SENSOR_PIN)

    sensor_data = {
        "temperature": round(temperature, 2),
        "co_level": mq7_value,
        "soil_moisture": soil_moisture,
        "steam_detected": bool(steam_status),
        "timestamp": time.strftime("%Y-%m-%d %H:%M:%S")
    }

    ref.set(sensor_data) # Update Firebase Database
    print("Uploaded:", sensor_data)

while True:
    upload_sensor_data()
    time.sleep(5) # Upload data every 5 seconds

```

◆ 5. Firebase Web App (Mobile Visualization)

Create Firebase-Hosted Web Dashboard

1. Go to Firebase Console → Authentication
2. Enable Anonymous Sign-In
3. Go to Realtime Database → Enable Read Access

◆ Create `index.html` (Real-Time Data Display)

html

```
<!DOCTYPE html>
<html>
<head>
  <title>Air Quality Monitor</title>
  <script src="https://www.gstatic.com/firebasejs/9.0.2/firebase-app.js"></script>
  <script src="https://www.gstatic.com/firebasejs/9.0.2/firebase-database.js">
</script>
</head>
<body>
  <h1>Air Quality Monitor</h1>
  <p><b>Temperature:</b> <span id="temperature">Loading...</span> °C</p>
  <p><b>CO Level:</b> <span id="co">Loading...</span></p>
  <p><b>Soil Moisture:</b> <span id="soil">Loading...</span></p>
  <p><b>Steam Detected:</b> <span id="steam">Loading...</span></p>

  <script>
    const firebaseConfig = {
      apiKey: "your-api-key",
      authDomain: "your-project.firebaseio.com",
      databaseURL: "https://your-project.firebaseio.com",
      projectId: "your-project",
      storageBucket: "your-project.appspot.com",
      messagingSenderId: "your-messaging-id",
      appId: "your-app-id"
    };

    firebase.initializeApp(firebaseConfig);
    const dbRef = firebase.database().ref("/sensor_data");

    dbRef.on("value", (snapshot) => {
      const data = snapshot.val();
      document.getElementById("temperature").innerText = data.temperature;
      document.getElementById("co").innerText = data.co_level;
      document.getElementById("soil").innerText = data.soil_moisture;
      document.getElementById("steam").innerText = data.steam_detected ? "Yes"
: "No";
    });
  </script>
```

```
</body>  
</html>
```

◆ 6. Deploy Web App to Firebase Hosting (for Mobile Access)

1. Install Firebase CLI:

```
bash  
  
npm install -g firebase-tools
```

2. Login to Firebase:

```
bash  
  
firebase login
```

3. Initialize Firebase Hosting:

```
bash  
  
firebase init
```

- Select **Hosting**
- Choose the project
- Set `public` folder to where `index.html` is stored

4. Deploy Web App:

```
bash  
  
firebase deploy
```

5. Access from Mobile: Open the Firebase Hosting URL in your browser

Summary

- ✓ Raspberry Pi reads sensor data (MQ7, Soil Moisture, LM35, Steam Sensor)
- ✓ Data is sent to Firebase Realtime Database
- ✓ Firebase Web App displays live data
- ✓ Access the dashboard from mobile via Firebase Hosting

Would you like **notifications on mobile** for high CO levels? 