Universidad Mariano Galvez De Guatemala

Inteligencia Artificial

Ing. Jonathan Cuque

Manual técnico proyecto final

|  |  |
| --- | --- |
| Derihan Arnoldo Guevara Solorzano | 5390-16-19827 |
| Héctor Eli Contreras Hernández | 5390-22-12755 |
|  |  |
|  |  |

Indice

Contenido

[Introducción 2](#_Toc199613738)

[Manual Técnico de entrenamiento de calidad del aire 3](#_Toc199613739)

[Calibración del sensor – calculo de Ro 4](#_Toc199613740)

[Entrenamiento del modelo 5](#_Toc199613741)

[Conclusión 8](#_Toc199613742)

# Introducción

En la actualidad, la contaminación del aire representa uno de los principales problemas ambientales que afectan la salud humana y el medio ambiente. Para abordar esta problemática, es fundamental contar con sistemas de monitoreo de calidad del aire que permitan detectar y predecir los niveles de contaminantes de forma accesible y eficiente.

Este proyecto se enfoca en la recolección y análisis de datos ambientales utilizando sensores de bajo costo, específicamente el sensor **MQ135** para medir la calidad del aire, y el sensor **DHT11** para obtener lecturas de temperatura y humedad. A través de un sistema basado en Arduino, se capturan los datos en tiempo real, los cuales luego son utilizados en un entorno de programación en **Python** para entrenar un modelo de **regresión lineal**.

El objetivo principal es predecir los valores del sensor MQ135 (indicativo de gases contaminantes como amoníaco, alcohol, benceno y CO₂) a partir de las variables ambientales temperatura y humedad. Esto permite obtener una estimación de la calidad del aire sin necesidad de realizar una lectura directa del sensor, lo cual puede ser útil para análisis históricos, mantenimiento predictivo o implementación de alertas automáticas.

El proceso general del proyecto consta de las siguientes etapas:

1. **Calentamiento del sensor MQ135:** Fase inicial necesaria para estabilizar el sensor antes de realizar lecturas confiables.
2. **Calibración:** Estimación de la resistencia base del sensor en un ambiente limpio para obtener mayor precisión.
3. **Lectura de datos:** Recolección de datos de temperatura, humedad y calidad del aire desde Arduino.
4. **Entrenamiento del modelo:** Análisis de los datos utilizando técnicas de *Machine Learning* para predecir el valor del sensor MQ135 a partir de las otras dos variables.

Este enfoque permite construir una herramienta accesible y educativa que combina electrónica con inteligencia artificial para aplicaciones en monitoreo ambiental.

# Manual Técnico de entrenamiento de calidad del aire

ETAPA 1: HARDWARE Y LECTURAS (CÓDIGO ARDUINO)

#include <DHT.h>

#define DHTPIN 2 // Pin digital conectado al sensor DHT

#define DHTTYPE DHT11 // Tipo de sensor

#define MQ135PIN A0 // Pin analógico para el MQ135

DHT dht(DHTPIN, DHTTYPE);

void setup() {

Serial.begin(9600); // Inicia comunicación serial

dht.begin(); // Inicializa el sensor DHT

}

void loop() {

float temperatura = dht.readTemperature(); // Lee temperatura

float humedad = dht.readHumidity(); // Lee humedad

int mq135 = analogRead(MQ135PIN); // Lee el valor analógico del sensor MQ-135

if (isnan(temperatura) || isnan(humedad)) {

Serial.println("Error al leer el sensor DHT11");

return; // Sale del loop si hay error

}

// Imprime los datos en formato CSV para guardarlos en un archivo

Serial.print(temperatura, 2);

Serial.print(",");

Serial.print(humedad, 2);

Serial.print(",");

Serial.println(mq135);

delay(1000); // Espera 1 segundo antes de la siguiente lectura

}

# Calibración del sensor – calculo de Ro

Determinar el valor de **Ro**, la resistencia del sensor en aire limpio. Es fundamental para interpretar correctamente las lecturas del MQ-135.

**Fórmula general:**

Interfaz de usuario gráfica, Diagrama, Aplicación

El contenido generado por IA puede ser incorrecto.

RLRs=(Vout5.0−Vout​)×RL Ro=Rs/Ratio\_limpioRo = Rs / Ratio\\_limpioRo=Rs/Ratio\_limpio

* RL: resistencia de carga (generalmente 10 kΩ)
* Ratio\_limpio: para el MQ-135 es típicamente 3.6 a 10 en aire limpio. Se puede usar 3.6 como valor común.

Lectura y registro de datos

Recolectar datos del ambiente: **Temperatura**, **Humedad** y **MQ-135** (Rs/Ro), para posteriormente entrenar un modelo de Machine Learning.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Fecha | Hora | Temperatura | Humedad | MQ135 | Rs\_Ro | Ubicación |
| 2025-05-31 | 14:00 | 25.4 | 65.0 | 314.2 | 2.85 | Casa |

# Entrenamiento del modelo

Usar los datos obtenidos para predecir los valores del sensor MQ-135 a partir de temperatura y humedad.

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error, r2\_score

import matplotlib.pyplot as plt

# 1. Cargar solo columnas necesarias

columnas\_utilizadas = ['Temperatura\_C', 'Humedad', 'MQ135']

df = pd.read\_csv('datos\_calidad\_aire.csv', usecols=columnas\_utilizadas, encoding='latin1')

# 2. Convertir a números y limpiar datos faltantes

df = df.apply(pd.to\_numeric, errors='coerce')

df\_sin\_nan = df.dropna()

# 3. Verificar que haya datos válidos

if df\_sin\_nan.empty:

print("No hay suficientes datos para entrenar el modelo.")

exit()

# 4. Definir variables independientes (X) y dependiente (y)

X = df\_sin\_nan[['Temperatura\_C', 'Humedad']]

y = df\_sin\_nan['MQ135']

# 5. Dividir en datos de entrenamiento y prueba

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# 6. Entrenar el modelo

modelo = LinearRegression()

modelo.fit(X\_train, y\_train)

# 7. Predecir resultados y evaluar

y\_pred = modelo.predict(X\_test)

print("Evaluación del modelo:")

print("MSE:", mean\_squared\_error(y\_test, y\_pred))

print("R²:", r2\_score(y\_test, y\_pred))

# 8. Graficar resultados

plt.figure(figsize=(10, 5))

plt.scatter(y\_test, y\_pred)

plt.plot([y\_test.min(), y\_test.max()], [y\_test.min(), y\_test.max()], 'k--', lw=2)

plt.xlabel("Valor Real de MQ135")

plt.ylabel("Valor Predicho de MQ135")

plt.title("Predicción de MQ135 vs. Valor Real")

plt.grid(True)

plt.tight\_layout()

plt.show()

# Conclusión

Este flujo garantiza que el sensor esté operando en condiciones estables, calibrado correctamente, y que los datos recolectados puedan usarse para crear modelos predictivos confiables.