Imagen que contiene Calendario

El contenido generado por IA puede ser incorrecto.**UNIVERSIDAD MARIANO GÁLVEZ DE GUATEMALA**

**Campus Villa Nueva**

**Facultad de ingeniería en sistemas de información y ciencias de la computación**

**Inteligencia Artificial**

# **Proyecto Final**

|  |  |
| --- | --- |
| **INTEGRANTES** | **CARNET** |
| Derihan Arnoldo Guevara Solorzano | 5390-16-19837 |
| Héctor Eli Contreras Hernandez | 5390-22-12755 |
| Jorge Ricado Ubico Brooks | 5390-09-15892 |
| Carlos Daniel Paredes Grijalva | 5390-18-744 |
|  |  |
|  |  |

**Sección “A”**

**Mayo 2025**

## Introducción

El fin de este proyecto es poder realizar un programa con inteligencia artificial y Arduino para medir la calidad del aire. No solo el medirla también entrenar al sistema para que pueda realizar la medición del aire de una mejor manera y con predicción por cada ubicación. Se implemento un censor MQ135 el cual su función es la medición de la calidad del aire este a su vez conectado con un Arduino que se conecta por un serial al programa de Python.

Índice

[**Proyecto Final** 1](#_Toc199612825)

[Introducción 2](#_Toc199612826)

[Código y Funcionamiento 4](#_Toc199612827)

[Entrenamiento 4](#_Toc199612828)

[Lecturas de datos 6](#_Toc199612829)

[Calibración 9](#_Toc199612830)

[Calentamiento del sensor 11](#_Toc199612831)

[Arduino 14](#_Toc199612832)

[Circuitos y Diagrama de Conexión 15](#_Toc199612833)

[Enlace Git 16](#_Toc199612834)

[Reporte Power Bi 16](#_Toc199612835)

[URL 16](#_Toc199612836)

## Código y Funcionamiento

### Entrenamiento

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

# Cargar solo las columnas necesarias y ajustar nombres

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

try:

df = pd.read\_csv('datos\_calidad\_aire.csv', header=0, names=[

'ID', 'Fecha', 'Hora', 'Temperatura\_C', 'Humedad',

'MQ135', 'Rs\_Ro', 'Calidad\_Aire\_ICA', 'Segundos\_Desde\_Inicio', 'Ubicacion'

], usecols=['Temperatura\_C', 'Humedad', 'MQ135'], encoding='latin1')

except FileNotFoundError:

print("Error: El archivo 'datos\_calidad\_aire.csv' no fue encontrado.")

exit()

# Convertir a tipo numérico, forzando errores a NaN

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

# Eliminar filas que contengan al menos un NaN

df\_sin\_nan = df.dropna()

if df\_sin\_nan.empty:

print("Error: No hay suficientes datos válidos para entrenar el modelo después de eliminar filas con NaN.")

exit()

# Definir variables independientes y dependientes

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

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

# Dividir en entrenamiento y prueba

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

# Entrenar el modelo

modelo = LinearRegression()

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

# Realizar predicciones

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

# Evaluar el modelo

print(" Evaluación del modelo:")

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

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

# 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()

# Otra forma de graficar (series de tiempo) si los datos tienen un orden temporal

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

# plt.plot(y\_test.values, label='Real', marker='o')

# plt.plot(y\_pred, label='Predicción', marker='x')

# plt.title("Predicción de MQ135 (PM2.5 simulado)")

# plt.xlabel("Muestras")

# plt.ylabel("Valor MQ135")

# plt.legend()

# plt.grid(True)

# plt.tight\_layout()

# plt.show()

### Lecturas de datos

import serial

import csv

import time

import os

import json

from datetime import datetime

# UBICACIÓN FIJA

ubicacion = "oficina"

if not os.path.exists("ro\_config.json"):

print("No se encontró el archivo de calibración (ro\_config.json). Ejecuta primero la calibración.")

exit()

with open("ro\_config.json", "r") as f:

Ro = json.load(f)["Ro"]

print(f"Ro cargado: {Ro:.2f}")

# Configurar puerto

arduino = serial.Serial('COM3', 9600)

time.sleep(2)

filename = 'datos\_calidad\_aire.csv'

archivo\_nuevo = not os.path.exists(filename)

def interpretar\_calidad\_ica\_calibrado(relacion):

if relacion > 5.0:

return "Morado (Muy dañino a la salud)"

elif 3.5 <= relacion <= 5.0:

return "Rojo (Dañino a la salud)"

elif 2.0 <= relacion < 3.5:

return "Naranja (Dañino a la salud para grupos sensibles)"

elif 1.0 <= relacion < 2.0:

return "Amarillo (Moderado)"

else:

return "Verde (Bueno)"

# Iniciar archivo CSV

with open(filename, mode='a', newline='') as file:

writer = csv.writer(file)

if archivo\_nuevo:

writer.writerow([

'ID', 'Fecha', 'Hora', 'Temperatura\_C', 'Humedad',

'MQ135', 'Rs\_Ro', 'Calidad\_Aire\_ICA', 'Segundos\_Desde\_Inicio', 'Ubicacion'

])

print(" Recolectando datos... Ctrl+C para detener.")

id\_muestra = 1

inicio = time.time()

try:

while True:

linea = arduino.readline().decode('utf-8').strip()

if linea.count(',') == 2:

datos = linea.split(',')

try:

temp = float(datos[0])

hum = float(datos[1])

mq = float(datos[2])

relacion = round(mq / Ro, 2)

calidad\_ica = interpretar\_calidad\_ica\_calibrado(relacion)

ahora = datetime.now().strftime('%Y-%m-%d %H:%M:%S')

segundos\_transcurridos = round(time.time() - inicio, 2)

writer.writerow([

id\_muestra,

ahora.split(' ')[0],

ahora.split(' ')[1],

temp, hum, mq, relacion, calidad\_ica, segundos\_transcurridos, ubicacion

])

print(f"[{ahora}] Temp: {temp}°C | Hum: {hum}% | MQ: {mq} | Rs/Ro: {relacion:.2f} | Calidad Aire (ICA): {calidad\_ica} | Ubicación: ({ubicacion})")

id\_muestra += 1

except Exception as e:

print(" Error en conversión de datos:", e)

except KeyboardInterrupt:

print(f"Lectura detenida. Datos guardados en {filename}")

 arduino.close()

### Calibración

import serial

import time

import json

import os

# Configuración del puerto serie

arduino = serial.Serial('COM3', 9600)

time.sleep(2)

# Calibrar durante 15 segundos

tiempo\_calibracion = 15

valores\_calibracion = []

print(f"Iniciando calibración durante {tiempo\_calibracion} segundos...")

inicio\_calibracion = time.time()

while time.time() - inicio\_calibracion < tiempo\_calibracion:

try:

linea = arduino.readline().decode('utf-8').strip()

if linea.count(',') == 2:

datos = linea.split(',')

mq\_valor = float(datos[2])

valores\_calibracion.append(mq\_valor)

print("Lectura:", mq\_valor)

except:

continue

arduino.close()

if len(valores\_calibracion) == 0:

print("No se obtuvieron datos.")

exit()

Ro = sum(valores\_calibracion) / len(valores\_calibracion)

print(f" Ro calibrado: {Ro:.2f}")

# Guardar en archivo

with open('ro\_config.json', 'w') as f:

json.dump({"Ro": Ro}, f)

print("Ro guardado en ro\_config.json")

### Calentamiento del sensor

import serial

import time

from datetime import datetime

# Configura el puerto COM

arduino = serial.Serial('COM3', 9600)

time.sleep(2)

# Calentamiento (10 minutos = 600 segundos)

tiempo\_calentamiento = 600

print(f"Calentando sensor durante {tiempo\_calentamiento // 60} minutos...")

inicio\_calentamiento = time.time()

while time.time() - inicio\_calentamiento < tiempo\_calentamiento:

tiempo\_restante = int(tiempo\_calentamiento - (time.time() - inicio\_calentamiento))

print(f"Tiempo restante para iniciar calibración: {tiempo\_restante} segundos", end='\r')

time.sleep(1)

print("\nCalentamiento finalizado. Iniciando calibración...")

# Calibración

tiempo\_calibracion = 15

valores\_calibracion = []

inicio\_calibracion = time.time()

while time.time() - inicio\_calibracion < tiempo\_calibracion:

linea = arduino.readline().decode('utf-8').strip()

if linea.count(',') == 2:

datos = linea.split(',')

try:

mq\_valor = float(datos[2])

valores\_calibracion.append(mq\_valor)

print(f"Valor calibración: {mq\_valor}")

except:

continue

if not valores\_calibracion:

print("No se obtuvieron valores válidos para calibración.")

arduino.close()

exit()

Ro = sum(valores\_calibracion) / len(valores\_calibracion)

print(f"Calibración completada. Ro estimado: {Ro:.2f}")

# Función de interpretación

def interpretar\_calidad\_relativa(mq):

relacion = mq / Ro

if relacion > 6.0:

return "Excelente"

elif relacion > 4.0:

return "Buena"

elif relacion > 2.5:

return "Regular"

elif relacion > 1.5:

return "Mala"

else:

return "Muy Contaminado"

print("Iniciando lectura continua... Ctrl+C para salir.")

try:

while True:

linea = arduino.readline().decode('utf-8').strip()

if linea.count(',') == 2:

datos = linea.split(',')

try:

temp = float(datos[0])

hum = float(datos[1])

mq = float(datos[2])

rs\_ro = mq / Ro

calidad = interpretar\_calidad\_relativa(mq)

ahora = datetime.now().strftime('%Y-%m-%d %H:%M:%S')

print(f"[{ahora}] Temp: {temp}°C | Hum: {hum}% | MQ: {mq} | Rs/Ro: {rs\_ro:.2f} | Calidad: {calidad}")

except:

continue

except KeyboardInterrupt:

print("\nLectura finalizada por el usuario.")

 arduino.close()

### 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);

dht.begin();

}

void loop() {

float temperatura = dht.readTemperature();

float humedad = dht.readHumidity();

int mq135 = analogRead(MQ135PIN); // Lectura del MQ135

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

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

return;

}

Serial.print(temperatura, 2);

Serial.print(",");

Serial.print(humedad, 2);

Serial.print(",");

Serial.println(mq135);

delay(1000); // Espera 2 segundos

## Circuitos y Diagrama de Conexión

Un conjunto de letras negras en un fondo blanco

El contenido generado por IA puede ser incorrecto.Un circuito electrónico

El contenido generado por IA puede ser incorrecto.

Imagen que contiene dispositivo

El contenido generado por IA puede ser incorrecto. Imagen que contiene circuito

El contenido generado por IA puede ser incorrecto.

Un cable conectado

El contenido generado por IA puede ser incorrecto.

## Enlace Git

## Reporte Power Bi

Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

### URL

<https://drive.google.com/file/d/1fTV_vm2HPf2pSWl1yoPKYdxCXxohp6hp/view?usp=sharing>