

Universidad Mariano Gálvez De Guatemala
Facultad De Ingeniería En Sistemas
Inteligencia Artificial – Sección B
Ing. Jonathan Josué García Cuque



PROYECTO FINAL

INTEGRANTES DEL GRUPO#2

Jose Andre Mazariegos Aceytuno 5390-20-3221

Jorge Mario Garcia Santizo 5390-21-7603

Jerson Armando Ramirez Perez 5390-20-9608

David Emanuel Morales Mijangos 55390-21-12599

Medein Esdras Ramirez 5390-21-12599

Jorge Francisco Santos 5390-21-1308

Contenido

INTRODUCCION	3
CONFIGURACION EN ARDUINO	4
CONFIGURACION EN PYTHON	5
EXPLICACION DEL CODIGO DE ARDUINO	7
EXPLICACION DEL CODIGO DE PYTHON	8
ANALISIS DE DATOS EN POWER BI	10
CONCLUSIONES	13
FOTOGRAFIAS DEL CIRCUITO Y SENSORES INSTALADOS	14



INTRODUCCION

La calidad del aire es un factor determinante para la salud pública y el bienestar ambiental, especialmente en contextos urbanos donde la concentración de contaminantes como el material particulado fino (PM2.5) puede superar los límites recomendados. Con el objetivo de proporcionar una herramienta de monitoreo y análisis accesible, este proyecto propone el desarrollo de un sistema predictivo de calidad del aire que integra sensores físicos, algoritmos de aprendizaje automático y visualización avanzada de datos.

El sistema se basa en la recopilación de datos en tiempo real mediante una plataforma Arduino equipada con sensores DHT11 para temperatura y humedad, y MQ-135 para la detección de gases contaminantes. Estos datos se envían cada 10 segundos a un script en Python a través del puerto serial, donde son procesados por un modelo de Machine Learning entrenado previamente para predecir los niveles de PM2.5. El modelo, construido utilizando un algoritmo de regresión lineal, se entrena con registros históricos que incluyen variables ambientales como temperatura, humedad, concentración de CO₂ y valores del sensor MQ-135.

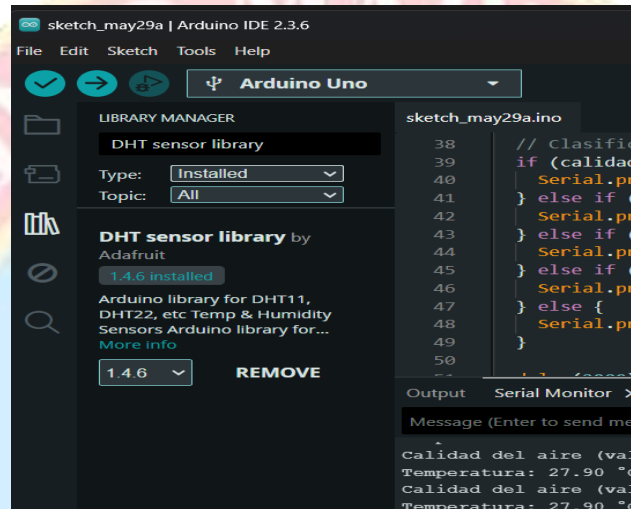
Para facilitar la interpretación de los resultados, se utiliza Power BI como herramienta de visualización, donde se representan mapas de calor que identifican zonas con mayor contaminación, gráficos de líneas que muestran la evolución temporal de los niveles de PM2.5, e indicadores de riesgo ambiental que alertan sobre condiciones adversas para la salud.

Este enfoque permite no solo monitorear el estado actual del aire, sino también anticipar comportamientos futuros, lo que lo convierte en una solución práctica y escalable para aplicaciones educativas, urbanas o comunitarias en el contexto de la sostenibilidad ambiental.

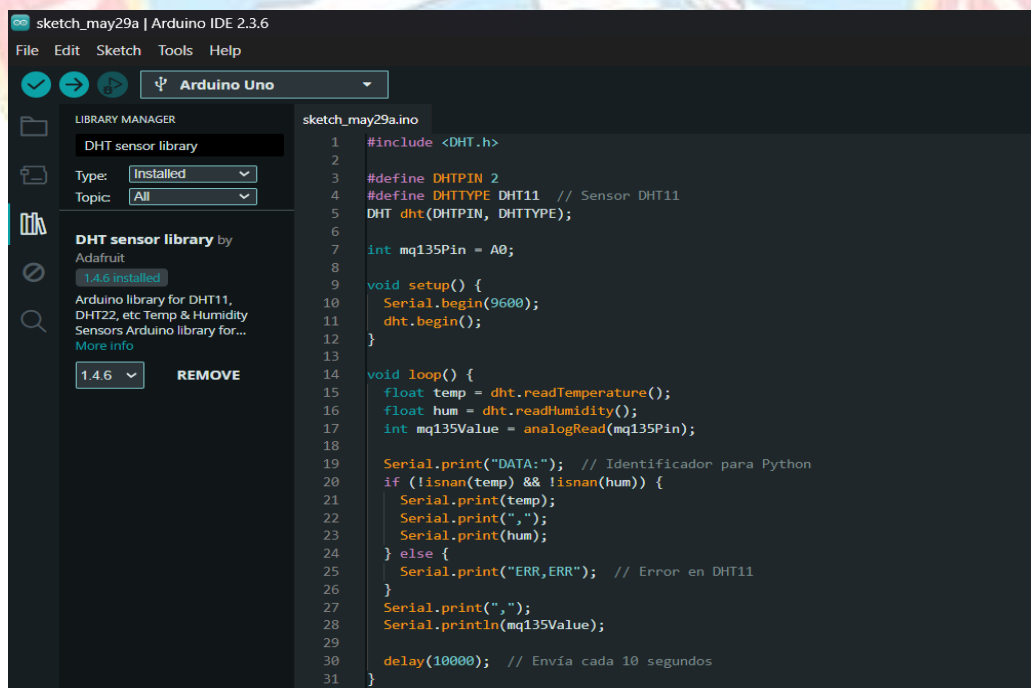
CONFIGURACION EN ARDUINO

El primer paso para la lectura de los sensores es crear el programa en Arduino y muy importante instalar la siguiente librería la cual va a interpretar los datos de nuestro sensor DHT11/DHT22 para temperatura y humedad.

La librería se llama DHT sensor library y es creada específicamente para este tipo de sensores.



Despues cargamos el programa funcional a Arduino para lograr enviar datos a Python, el código realizado en Arduino es el siguiente:



CONFIGURACION EN PYTHON

Procedemos a agregar el programa a Python para la lectura y predicciones para posteriormente pasarlo a power bi.

Para iniciar con nuestro código es necesario instalar las librerías necesarias en Python estos los instalamos desde la terminal dentro de la carpeta donde se encuentra el archivo Python del proyecto.

Se instalaron las siguientes librerías:

1. `py -m ensurepip --upgrade`
2. `py -m pip install --upgrade pip`
3. `py -m pip install pyserial pandas scikit-learn joblib`
4. `python -m pip install pandas scikit-learn pyserial joblib matplotlib`

```
PROBLEMAS  TERMINAL  SALIDA  PUERTOS  CONSOLA DE DEPURACIÓN

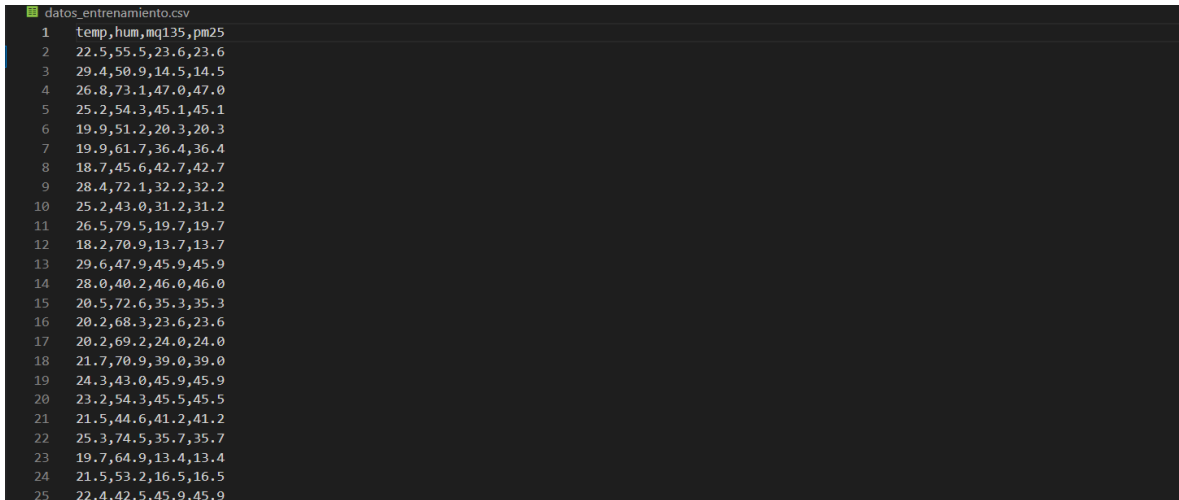
11.6/11.6 MB 3.0 MB/s eta 0:00:00
Downloading scikit_learn-1.6.1-cp310-cp310-win_amd64.whl (11.1 MB)
11.1/11.1 MB 2.0 MB/s eta 0:00:00
Downloading joblib-1.5.1-py3-none-any.whl (307 kB)
Downloading pytz-2025.2-py2.py3-none-any.whl (509 kB)
Downloading threadpoolctl-3.6.0-py3-none-any.whl (18 kB)
Downloading tzdata-2025.2-py2.py3-none-any.whl (347 kB)
Installing collected packages: pytz, tzdata, threadpoolctl, joblib, scikit-learn, pandas
Successfully installed joblib-1.5.1 pandas-2.2.3 pytz-2025.2 scikit-learn-1.6.1 threadpoolctl-3.6.0 tzdata-2025.2
PS C:\Users\Jerson\Desktop\Proyecto Final IA>
```

Ya instaladas las librerías creamos el archivo para entrenar a nuestro modelo.

```
entrenar_modelo.py > ...
1  import pandas as pd
2  from sklearn.linear_model import LinearRegression
3  from sklearn.model_selection import train_test_split
4  import joblib
5
6  # Cargar datos simulados/históricos
7  df = pd.read_csv("C:/Users/Jerson/Desktop/Proyecto Final IA/datos_entrenamiento.csv")
8
9  # Entrenamiento
10 X = df[["temp", "hum", "mq135"]]
11 y = df["pm25"]
12
13 modelo = LinearRegression()
14 modelo.fit(X, y)
15
16 # Guardar modelo
17 joblib.dump(modelo, "C:/Users/Jerson/Desktop/Proyecto Final IA/modelo_pm25.pkl")
18
19 print("✅ Modelo entrenado y guardado correctamente.")
20
```

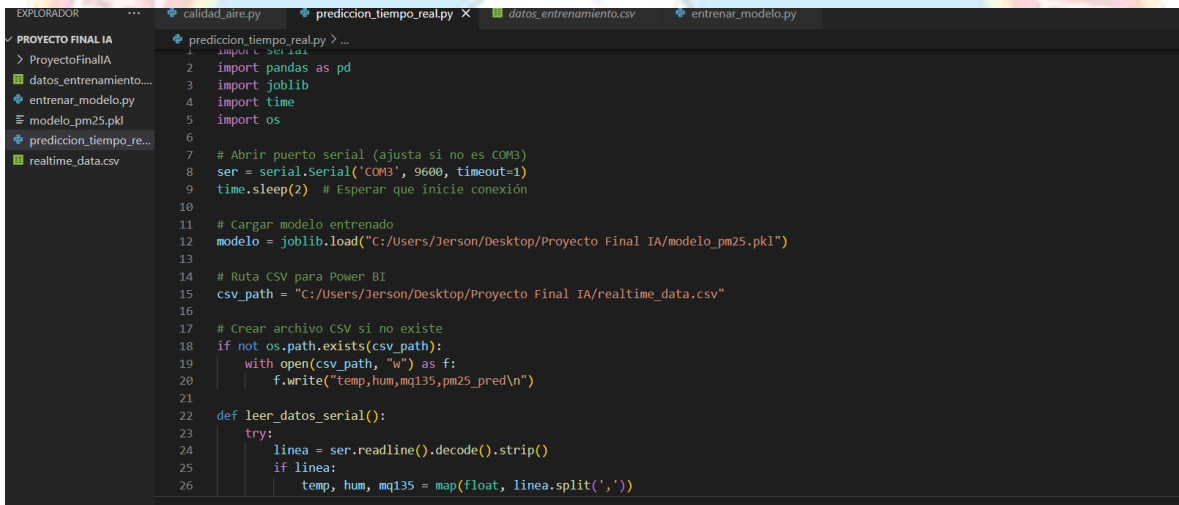
Este archivo necesita datos reales para poder entrenarse obtuvimos datos reales de internet y generamos un archivo csv para poder entrenar el modelo, punto importante en la web hay mediciones con pm2.5 solo hicimos la conversión para tratar de igualar las

mediciones con el sensor qm 135, se comparte visualización de los datos de entrenamiento:



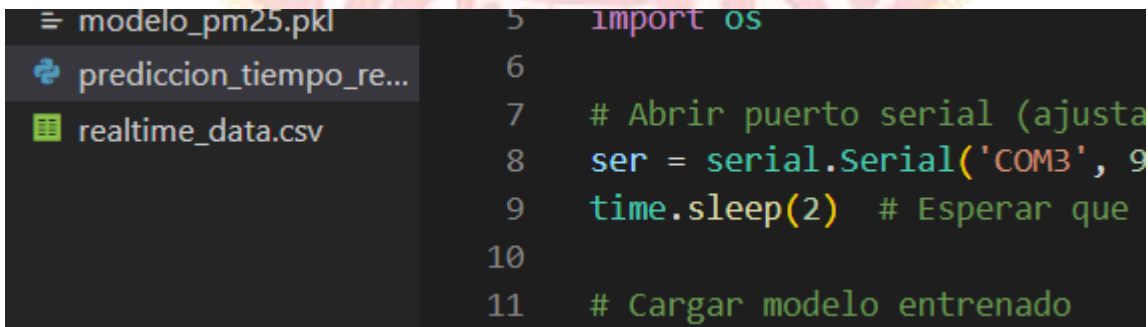
	temp	hum	mq135	pm25
1	22.5	55.5	23.6	23.6
2	29.4	50.9	14.5	14.5
3	26.8	73.1	47.0	47.0
4	25.2	54.3	45.1	45.1
5	19.9	51.2	20.3	20.3
6	19.9	61.7	36.4	36.4
7	18.7	45.6	42.7	42.7
8	28.4	72.1	32.2	32.2
9	25.2	43.0	31.2	31.2
10	26.5	79.5	19.7	19.7
11	18.2	70.9	13.7	13.7
12	29.6	47.9	45.9	45.9
13	28.0	40.2	46.0	46.0
14	20.5	72.6	35.3	35.3
15	20.2	68.3	23.6	23.6
16	20.2	69.2	24.0	24.0
17	21.7	70.9	39.0	39.0
18	24.3	43.0	45.9	45.9
19	23.2	54.3	45.5	45.5
20	21.5	44.6	41.2	41.2
21	25.3	74.5	35.7	35.7
22	19.7	64.9	13.4	13.4
23	21.5	53.2	16.5	16.5
24	22.4	42.5	45.9	45.9
25				

Despues creamos el archivo que va a obtener datos reales y nos generara el csv para usarlo en power bi.



```
1 import pandas as pd
2 import joblib
3 import time
4 import os
5
6 # Abrir puerto serial (ajusta si no es COM3)
7 ser = serial.Serial('COM3', 9600, timeout=1)
8 time.sleep(2) # Esperar que inicie conexión
9
10 # Cargar modelo entrenado
11 modelo = joblib.load("C:/Users/Jerson/Desktop/Proyecto Final IA/modelo_pm25.pkl")
12
13 # Ruta CSV para Power BI
14 csv_path = "C:/Users/Jerson/Desktop/Proyecto Final IA/realtime_data.csv"
15
16 # Crear archivo CSV si no existe
17 if not os.path.exists(csv_path):
18     with open(csv_path, "w") as f:
19         f.write("temp, hum, mq135, pm25_pred\n")
20
21 def leer_datos_serial():
22     try:
23         linea = ser.readline().decode().strip()
24         if linea:
25             temp, hum, mq135 = map(float, linea.split(','))
26             pm25_pred = modelo.predict([temp, hum, mq135])
27             with open(csv_path, "a") as f:
28                 f.write(f"{temp},{hum},{mq135},{pm25_pred}\n")
29     except Exception as e:
30         print(f"Error: {e}")
```

Al ejecutar este archivo nos creará uno llamado realtime.csv que nos servirá para el modelo en power bi.



```
5 import os
6
7 # Abrir puerto serial (ajusta
8 ser = serial.Serial('COM3', 9
9 time.sleep(2) # Esperar que
10
11 # Cargar modelo entrenado
```

EXPLICACION DEL CODIGO DE ARDUINO

1. Inclusión de bibliotecas y definición de pines

Se incluye la librería DHT.h que permite interactuar fácilmente con el sensor DHT11:

```
#include <DHT.h>
```

Se definen constantes para los pines conectados:

1. DHTPIN: Pin digital 2 conectado al DHT11.
2. DHTTYPE: Se indica que el sensor es del tipo DHT11.
3. MQ135PIN: Pin analógico A0 conectado al sensor MQ-135.

```
#define DHTPIN 2  
#define DHTTYPE DHT11  
#define MQ135PIN A0
```

2. Inicialización del sensor DHT

```
DHT dht(DHTPIN, DHTTYPE);
```

Se crea un objeto dht usando los parámetros definidos para el pin y el tipo de sensor.

3. Configuración inicial

```
void setup() {  
  Serial.begin(9600);  
  dht.begin();  
}
```

En la función setup():

Se inicializa la comunicación serial a 9600 baudios.

Se inicia el sensor DHT11 con dht.begin().

4. Bucle principal (loop)

```
float temp = dht.readTemperature();
```

```
float hum = dht.readHumidity();
```

Se leen la temperatura y la humedad del ambiente utilizando el sensor DHT11.

```
int mqRaw = analogRead(MQ135PIN);
```

Se obtiene una lectura analógica del sensor MQ-135, que mide gases contaminantes en el aire.

```
int mqScaled = map(mqRaw, 900, 950, 300, 600);
```

La lectura se ajusta al rango de 300 a 600 usando la función map(). Esto sirve para adaptar el valor a una escala más significativa para su visualización.

```
if (mqScaled < 300) mqScaled = 300;
```

```
if (mqScaled > 600) mqScaled = 600;
```

Se asegura que los valores ajustados no se salgan del rango definido (300–600).

5. Validación y envío de datos

```
if (isnan(temp) || isnan(hum)) {  
    Serial.println("Error leyendo DHT11");  
}
```

Se verifica si los valores de temperatura o humedad son inválidos (NaN, Not a Number). Si lo son, se imprime un mensaje de error.

```
else {  
    Serial.print(temp, 2);  
    Serial.print(",");  
    Serial.print(hum, 2);  
    Serial.print(",");  
    Serial.println(mqScaled);  
}
```

Si las lecturas son válidas, se envían al puerto serial los tres valores: temperatura, humedad y calidad del aire (escalada), separados por comas. Esto es útil para almacenar o visualizar los datos externamente (por ejemplo, en una gráfica o archivo CSV).

6. Espera entre mediciones

```
delay(10000);
```

Se espera 10 segundos antes de tomar una nueva medición para no saturar la lectura o el sistema que recibe los datos.

EXPLICACION DEL CODIGO DE PYTHON

Hicimos la importación de librerías

```
import serial, pandas as pd, joblib, time, os
```

- serial: Comunicación con Arduino vía puerto COM.
- pandas: Manipulación de datos en forma de tabla.
- joblib: Para cargar el modelo de Machine Learning.
- time: Manejo de pausas temporales.
- os: Verificación de existencia de archivos.

Inicializamos el puerto serial

```
ser = serial.Serial('COM3', 9600, timeout=1)
```

```
time.sleep(2)
```

1. Se abre el puerto COM3 a 9600 baudios.

2. Se espera 2 segundos para establecer conexión estable con Arduino.

Cargamos el modelo entrenado

```
modelo = joblib.load("../modelo_pm25.pkl")
```

Se carga el modelo previamente entrenado que predice los niveles de PM2.5 (material particulado fino) a partir de temperatura, humedad y MQ-135.

Inicializamos el archivo CSV

```
if not os.path.exists(csv_path):  
    with open(csv_path, "w") as f:  
        f.write("temp,hum,mq135,pm25_pred\n")
```

Si el archivo realtime_data.csv no existe, se crea con encabezados adecuados para importarlo en Power BI.

Esta es la función para leer desde Arduino

```
def leer_datos_serial():
```

Esta función hace lo siguiente:

1. Lee una línea del puerto serial.
2. Decodifica el texto y lo separa por comas.
3. Devuelve los valores de temperatura, humedad y MQ-135 como float.

El bucle principal es:

```
while True:  
    datos = leer_datos_serial()
```

Este bucle hace lo siguiente:

1. Lee datos del Arduino.
2. Aplica el modelo de predicción con `modelo.predict(...)`.
3. Escribe los datos y el valor predicho en el archivo CSV.
4. Muestra en consola la predicción de PM2.5.
5. Espera 10 segundos antes de repetir.

2. Entrenamiento del modelo (Python)

Este script entrena un modelo de regresión lineal para predecir niveles de PM2.5 en función de variables ambientales.

Explicación del código:

Carga de datos históricos

```
df = pd.read_csv("../datos_entrenamiento.csv")
```

Se cargan datos previamente recolectados que incluyen columnas como temp, hum, mq135 y pm25.

Separación de variables

- `X = df[["temp", "hum", "mq135"]]`
- `y = df["pm25"]`
- X: Variables independientes (entrada del modelo).
- y: Variable objetivo (salida esperada, nivel de PM2.5).

Entrenamiento del modelo

```
modelo = LinearRegression()  
modelo.fit(X, y)
```

Se crea y entrena un modelo de regresión lineal usando **scikit-learn**.

Guardado del modelo

```
joblib.dump(modelo, ".../modelo_pm25.pkl")
```

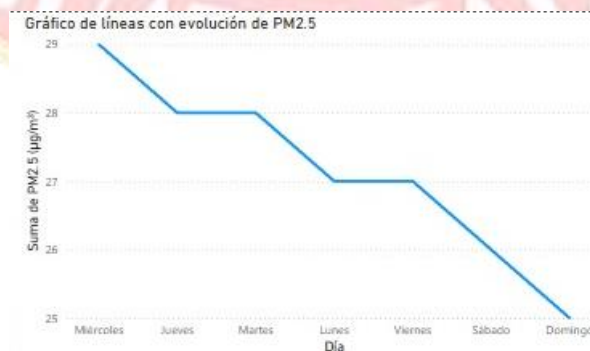
El modelo se guarda en un archivo .pkl para ser usado posteriormente en tiempo real.

ANALISIS DE DATOS EN POWER BI

Ya con la información generada por los sensores procedemos a analizarla en nuestro panel de visualización de Power BI, los graficos utilizados son:

Gráfico de líneas con evolución de PM2.5:

En este se puede apreciar el nivel de partículas en el aire según el día, esto quiere decir que en el eje y están los datos de los niveles de pm2.5 y en el eje x están los días de la semana, en esta grafica se puede ver el comportamiento de las partículas del día según los días en donde se aprecia del día con mayor y el día con menor partículas en el aire.

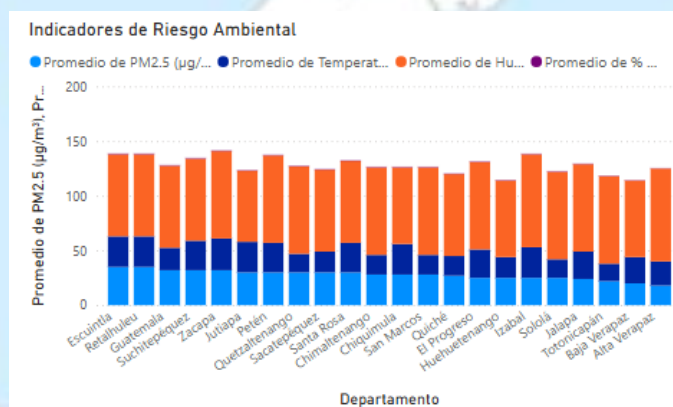


Para llevar a cabo este grafico se utilizó y ordeno la data de esta manera:

Día	PM2.5 ($\mu\text{g}/\text{m}^3$)
Lunes	27
Martes	28
Miércoles	29
Jueves	28
Viernes	27
Sábado	26
Domingo	25

Gráfico de Indicadores de Riesgo Ambiental:

En este grafico se pueden apreciar todos los valores según los 22 departamentos, acá se pueden apreciar los promedios de pm2.5, humedad y temperatura. Con estos valores se puede hacer un análisis y un comparativo con los departamentos.



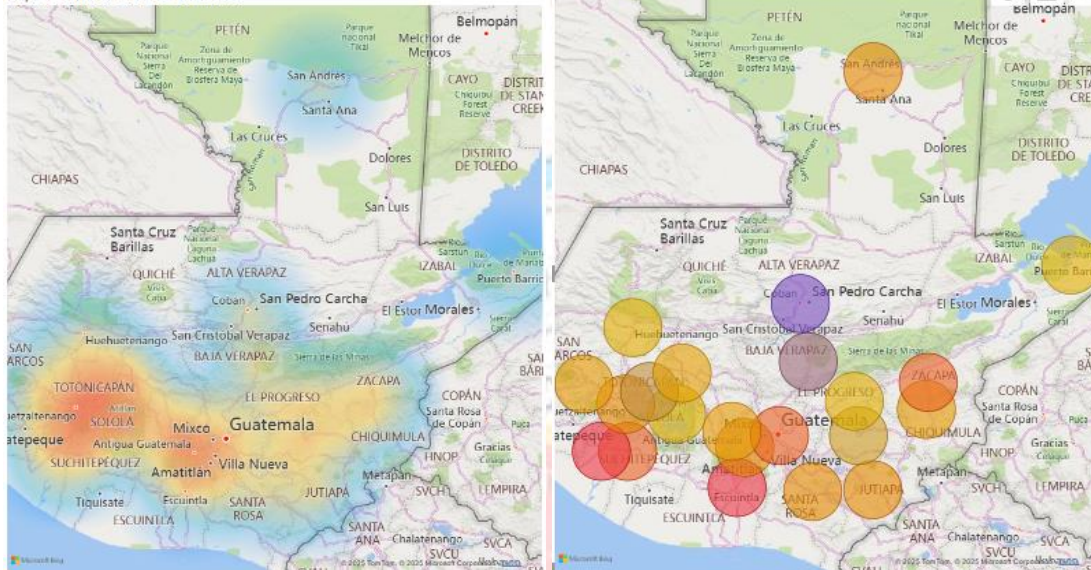
Para llevar a cabo este grafico se utilizó y ordeno la data de esta manera:

Departamento	Latitud	Longitud	Temperatura (°C)	Humedad (%)	PM2.5 (µg/m³)	% Contaminación
Guatemala	14.6349	-90.5069	20.5	75	32	0.55
El Progreso	14.8524	-90.0154	26	80	25	0.45
Sacatepéquez	14.5593	-90.734	19	75	30	0.5
Chimaltenango	14.6614	-90.8195	18	80	28	0.48
Escuintla	14.305	-90.7856	28	75	35	0.6
Santa Rosa	14.2793	-90.2887	27	75	30	0.5
Sololá	14.7723	-91.183	17	80	25	0.45
Totonicapán	14.9136	-91.3604	16	80	22	0.4
Quetzaltenango	14.8341	-91.518	17	80	30	0.5
Suchitepéquez	14.5347	-91.5041	27	75	32	0.55
Retalhuleu	14.5352	-91.6761	28	75	35	0.6
San Marcos	14.9623	-91.7903	18	80	28	0.48
Huehuetenango	15.319	-91.47	19	70	25	0.45
Quiché	15.0308	-91.1461	18	75	27	0.47
Baja Verapaz	15.1036	-90.3183	24	70	20	0.35
Alta Verapaz	15.4708	-90.37	22	85	18	0.3
Petén	16.9313	-89.892	27	80	30	0.5
Izabal	15.7146	-88.5926	28	85	25	0.45
Zacapa	14.9723	-89.5301	29	80	32	0.55
Chiquimula	14.8009	-89.5422	28	70	28	0.48
Jalapa	14.6345	-89.9887	25	80	24	0.42
Jutiapa	14.291	-89.895	28	65	30	0.5

Mapa de calor de zonas contaminadas:

En este mapa se utilizaron todos los valores de temperatura, humedad y pm2.5 para llevar a cabo un porcentaje de contaminación y poder identificar los departamentos mas afectados y plasmarlos en el mapa de Guatemala.

Mapa de calor de zonas contaminadas



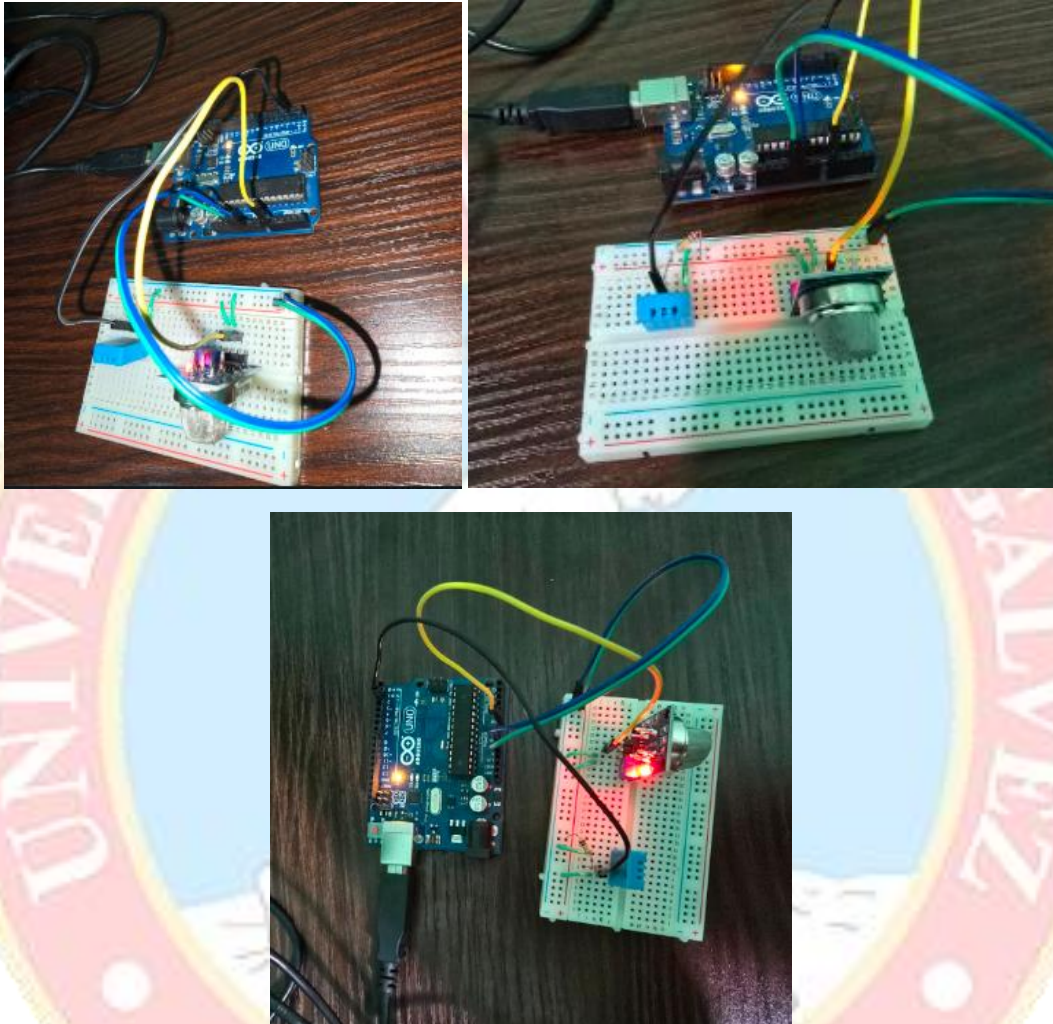
Para llevar a cabo este grafico se utilizó y ordeno la data de esta manera:

Departamento	Latitud	Longitud	Temperatura (°C)	Humedad (%)	PM2.5 (µg/m³)	% Contaminación
Guatemala	14.6349	-90.5069	20.5	75	32	0.55
El Progreso	14.8524	-90.0154	26	80	25	0.45
Sacatepéquez	14.5593	-90.734	19	75	30	0.5
Chimaltenango	14.6614	-90.8195	18	80	28	0.48
Escuintla	14.305	-90.7856	28	75	35	0.6
Santa Rosa	14.2793	-90.2887	27	75	30	0.5
Sololá	14.7723	-91.183	17	80	25	0.45
Totonicapán	14.9136	-91.3604	16	80	22	0.4
Quetzaltenango	14.8341	-91.518	17	80	30	0.5
Suchitepéquez	14.5347	-91.5041	27	75	32	0.55
Retalhuleu	14.5352	-91.6761	28	75	35	0.6
San Marcos	14.9623	-91.7903	18	80	28	0.48
Huehuetenango	15.319	-91.47	19	70	25	0.45
Quiché	15.0308	-91.1461	18	75	27	0.47
Baja Verapaz	15.1036	-90.3183	24	70	20	0.35
Alta Verapaz	15.4708	-90.37	22	85	18	0.3
Petén	16.9313	-89.892	27	80	30	0.5
Izabal	15.7146	-88.5926	28	85	25	0.45
Zacapa	14.9723	-89.5301	29	80	32	0.55
Chiquimula	14.8009	-89.5422	28	70	28	0.48
Jalapa	14.6345	-89.9887	25	80	24	0.42
Jutiapa	14.291	-89.895	28	65	30	0.5

CONCLUSIONES

1. Integración efectiva de hardware y software: El proyecto demostró que es posible desarrollar un sistema funcional de monitoreo ambiental en tiempo real utilizando sensores económicos conectados a una plataforma Arduino. La comunicación serial con Python permitió capturar datos de temperatura, humedad y calidad del aire de manera constante y confiable.
2. Aplicación práctica del aprendizaje automático: El uso de un modelo de regresión lineal entrenado con datos históricos permitió predecir niveles estimados de PM2.5 con base en variables ambientales locales. Esto valida la utilidad del aprendizaje automático como herramienta para anticipar condiciones de contaminación en tiempo real.
3. Visualización clara y útil de la información: La implementación de Power BI como plataforma de visualización facilitó la representación comprensible de los datos. Gráficos de líneas, indicadores de riesgo y mapas de calor ofrecieron una visión clara del estado y evolución de la calidad del aire, útil para la toma de decisiones o alertas preventivas.
4. Escalabilidad y adaptabilidad: El sistema desarrollado es fácilmente ampliable a otros contextos geográficos o entornos urbanos. Asimismo, puede integrarse con sensores adicionales o modelos de predicción más complejos, como redes neuronales, para mejorar la precisión y el alcance del análisis.
5. Contribución a la conciencia ambiental: Esta solución puede ser replicada como una herramienta educativa o comunitaria, fomentando la participación ciudadana en la vigilancia de la contaminación y promoviendo acciones informadas en favor del medio ambiente.

FOTOGRAFIAS DEL CIRCUITO Y DIAGRAMA DEL HARDWARE ARDUINO



Sistema predictivo de calidad de aire
GRUPO # 3

