**Universidad Autónoma de Tamaulipas**

**Facultad de Ingeniería Tampico**

A black background with a black square

Description automatically generated with medium confidenceA red and grey logo

Description automatically generated

**ASIGNATURA**

**Diseño Electrónico Basado en Sistemas Embebidos**

8vo. Semestre – Grupo “G”

2025 -1

**TRABAJO**

**Desarrollo de Prácticas y Proyectos**

**UNIDAD 2 – PROGRAMACIÓN BASICA**

**Docente:** Dr. García Ruiz Alejandro H.

|  |  |
| --- | --- |
| **Integrante del Equipo** | **Nivel de Participación** |
| Ortiz Doria Efrain Alejandro | 35 |
| Cristhian Michel Sandoval Vázquez | 35 |
| Luis Fernando Cruz Bonifacio | 15 |
| Adrián Segura Alonso | 15 |
| Total: | 100% |

# Índice

[Índice 1](#_Toc192426719)

[Repositorio(s) de Prácticas 2](#_Toc192426720)

[P6. Monitorio de Temperatura y Tiempo Real. 2](#_Toc192426721)

[Descrición de la practica 2](#_Toc192426722)

[Introducción 2](#_Toc192426723)

[Componentes para el desarrollo de la practica 3](#_Toc192426724)

[Desarrollo 5](#_Toc192426725)

[Conclusiones 10](#_Toc192426726)

# Repositorio(s) de Prácticas

|  |  |
| --- | --- |
| **Practica** | **Repositorio** |
| Practica 6 |  |

# P6. Monitorio de Temperatura y Tiempo Real.

## Descrición de la practica

Este proyecto implementa un sistema \*\*integrado hardware-software\*\* para el monitoreo y control térmico ambiental durante 24 horas continuas. Combina:

1. Captura de Datos:

- Sensor DHT11 para lecturas cada 2 minutos

- Transmisión serial de medianas horarias

2. Procesamiento en Tiempo Real:

- Limpieza de datos con detección adaptativa de outliers

- Suavizado exponencial (α=0.4)

- Control PID simplificado para activación de AC

3. Registro y Visualización:

- Archivo CSV con marca temporal

- Feedback visual mediante LED

## Introducción

En entornos industriales y domésticos, el monitoreo térmico confiable es esencial para:

- Mantener condiciones ambientales óptimas

- Reducir consumo energético

- Prevenir daños por sobrecalentamiento

Este sistema resuelve tres desafíos clave:

1. Calidad de Datos: Mitiga las limitaciones de sensores económicos mediante procesamiento estadístico

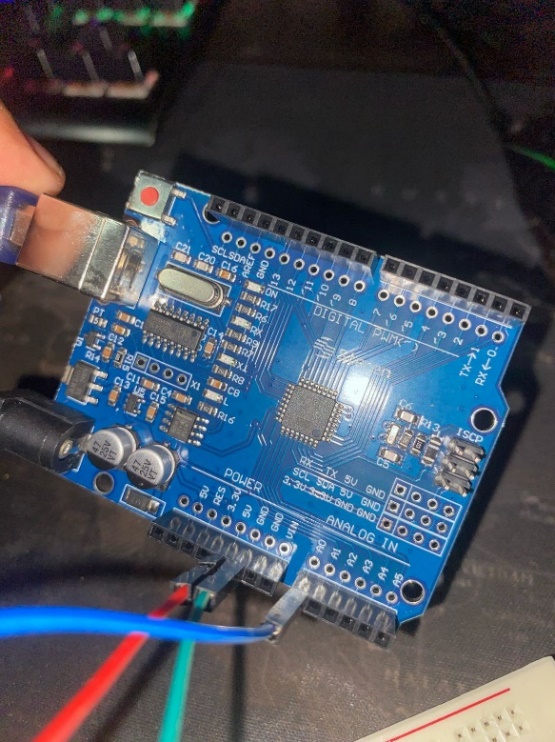
2. Toma de Decisiones Autónoma: Implementa lógica de control basada en umbrales dinámicos

3. Persistencia de Datos: Genera registros auditables para análisis posteriores

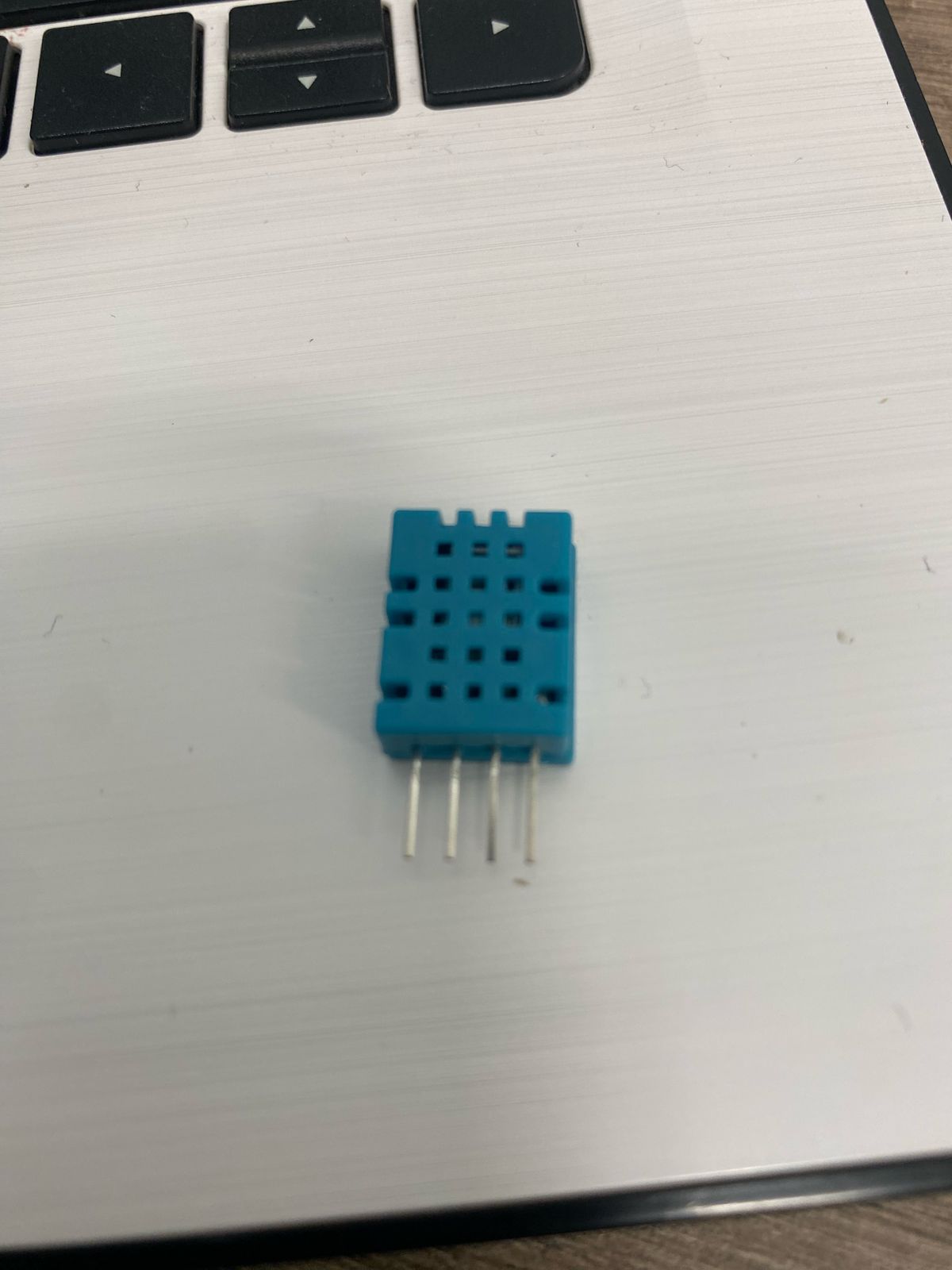
La integración Arduino-Python permite crear una solución escalable que combina lo mejor de ambos ecosistemas: baja latencia del firmware y poder analítico de Python.

## Componentes para el desarrollo de la practica

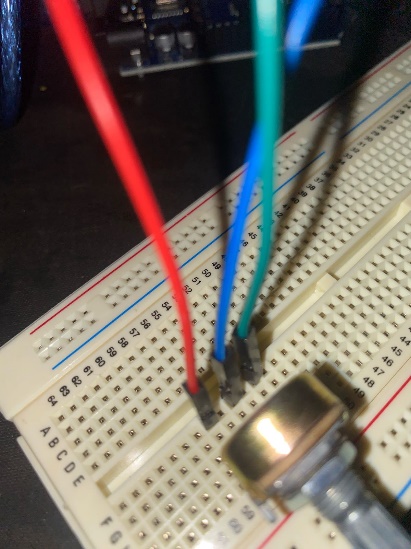
* **Componente 1.** Arduino UNO



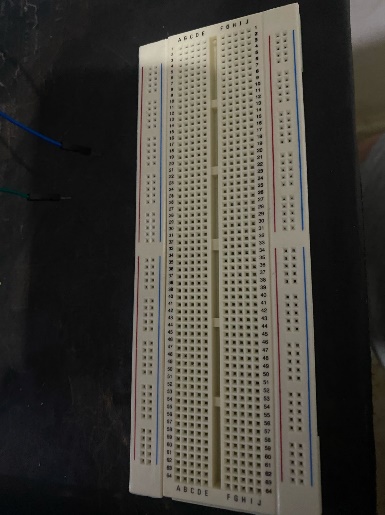
* **Componente 2**. Sensor Temperatura



* **Componente 3**. Cables



* **Componentes 4.** Protoboard



## Desarrollo

1.Configuración Hardware.

* DHT11:
  + VCC → 5V
  + DATA → Digital 2
  + GND → GND
* LED:
  + Ánodo → Digital 13
  + Cátodo → GND

Captura de pantalla de computadora

El contenido generado por IA puede ser incorrecto.

Captura de pantalla de computadora

El contenido generado por IA puede ser incorrecto.

1.**Clase Sistema Termico**



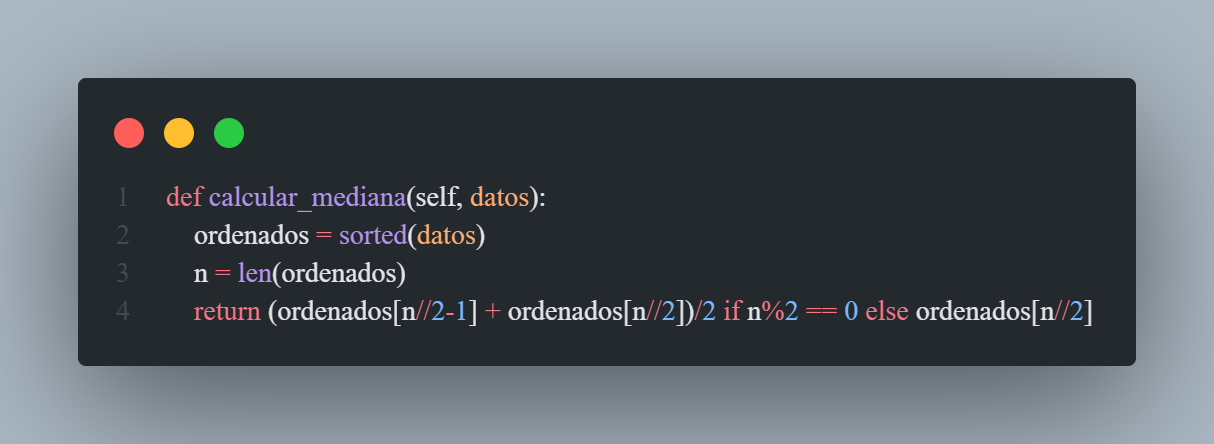
**Propósito**:

* Inicializa todos los componentes del sistema:
  + Parámetros de configuración (alpha, umbral)
  + Estructuras de datos (datos, suavizados)
  + Comunicación serial con Arduino
  + Sistema de registro en CSV

**Flujo de Inicialización**:

1. Establece valores iniciales para suavizado y umbral
2. Abre el puerto serial (COM3 a 115200 baudios)
3. Crea archivo CSV con encabezados
4. Espera 2 segundos para estabilizar la conexión

**2.Método calcular\_mediana**

****

**Función**:

* Calcula la mediana de una lista de valores numéricos

**Algoritmo**:

1. Ordena los datos de menor a mayor
2. Determina si la cantidad de elementos es par o impar
3. Para cantidad par: Promedia los dos valores centrales
4. Para cantidad impar: Toma el valor central

**Ejemplo**:

* Entrada: [23, 25, 24, 26, 22] → Ordenado: [22, 23, 24, 25, 26]
* Salida: 24 (tercer elemento)

**3. Método limpiar\_dato**

****

**Objetivo**:

* Identificar y corregir valores atípicos (outliers) usando:
  + **Ventana temporal**: Últimas 4 mediciones (4 horas)
  + **Método IQR**: Rango Intercuartílico

**Pasos**:

1. **Recolección de contexto**:
   * Crea lista con las últimas 4 mediciones + valor actual

contexto = self.datos[-ventana:] + [valor]

1. **Filtrado de datos válidos**:
   * Elimina valores None o inválidos

contexto\_limpio = [v for v in contexto if v is not None]

1. **Cálculo de límites IQR**:
   * Define rango "normal" de valores

q1 = ordenados[len(ordenados)//4] # Primer cuartil (25%)

q3 = ordenados[3\*len(ordenados)//4] # Tercer cuartil (75%)

iqr = q3 - q1 # Rango intercuartílico

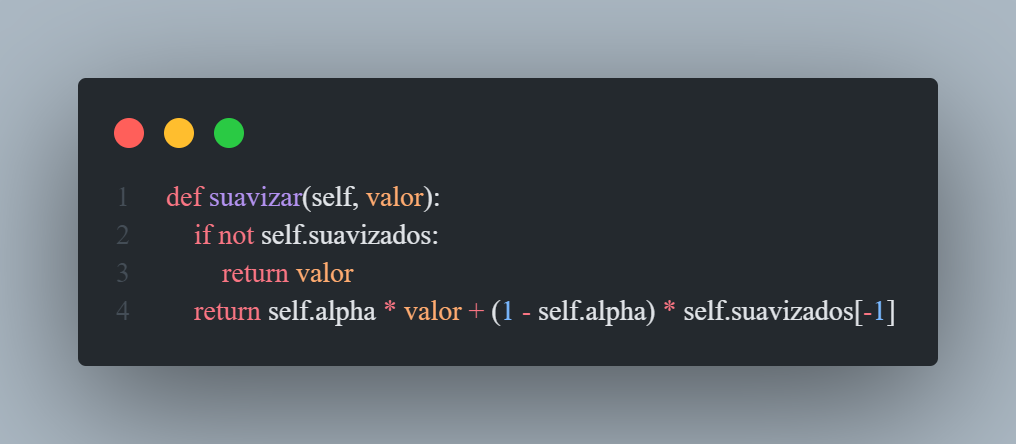
lim\_inf = q1 - 1.5\*iqr

lim\_sup = q3 + 1.5\*iqr

1. **Corrección de outliers**:
   * Si el valor está fuera de los límites, lo reemplaza por la mediana del contexto

return valor if lim\_inf <= valor <= lim\_sup else mediana

**4.Método suavizar**

****

**Propósito**:

* Aplicar **Suavizado Exponencial Simple (SES)** para:
  + Reducir variaciones bruscas
  + Mantener tendencias térmicas

**Fórmula**:

nuevo\_suavizado = α \* valor\_actual + (1 - α) \* suavizado\_anterior

**Parámetro Alpha (α=0.4)**:

* **α alto (0.4)**: Mayor peso al valor actual → Respuesta rápida
* **1-α (0.6)**: Peso al histórico → Estabilidad

**Ejemplo**:

* Valor actual: 25°C
* Suavizado anterior: 24°C
* Nuevo suavizado: 0.4\*25 + 0.6\*24 = 24.4°C

**5. Método controlar\_ac**

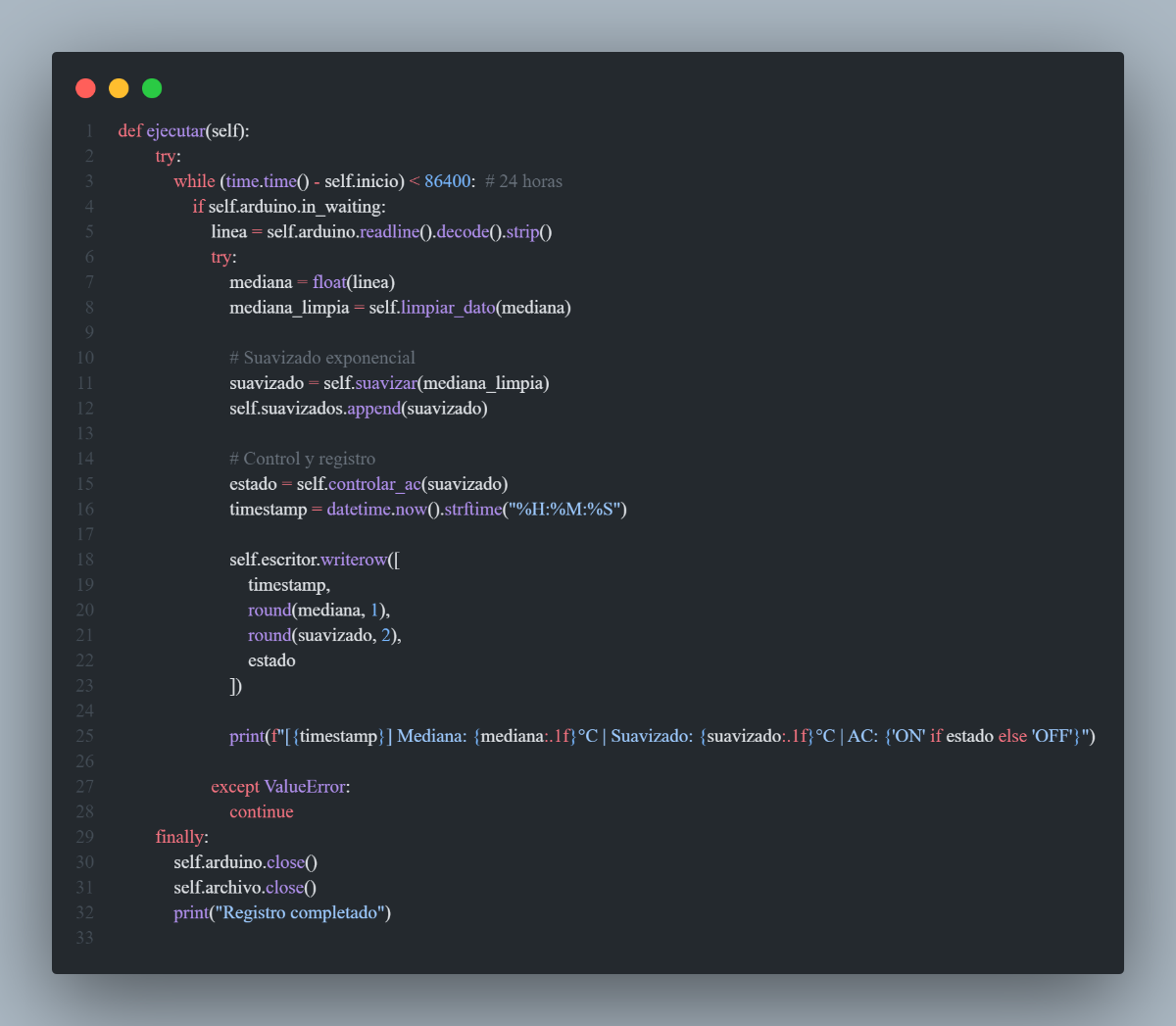
  
**Lógica de Control**:

* **Temperatura > 26°C**:
  + Envía 'L' por serial → Enciende LED/AC
* **Temperatura ≤ 26°C**:
  + Envía 'l' → Apaga LED/AC

**Comunicación Serial**:

* b'L'/b'l': Envía bytes (formato requerido por Arduino)
* Arduino interpreta estos caracteres para control físico

**6. Método ejecutar**



**Flujo Principal**:

1. **Bucle de 24 horas**:

while (time.time() - self.inicio) < 86400

* + Ejecuta durante exactamente 86,400 segundos (24h)

1. **Lectura Serial**:

if self.arduino.in\_waiting:

linea = self.arduino.readline().decode().strip()

* + Verifica datos disponibles en el puerto
  + Lee y decodifica la línea recibida

1. **Procesamiento en Cascada**:

A[Convertir a float] --> B[Limpieza de datos]

B --> C[Suavizado exponencial]

C --> D[Control AC]

D --> E[Registro CSV]

1. **Manejo de Errores**:

except ValueError:

continue

* + Ignora líneas no convertibles a float

1. **Cierre Seguro**:

finally:

self.arduino.close()

self.archivo.close()

* + Garantiza liberación de recursos

## Conclusiones

El sistema desarrollado demuestra que es posible implementar soluciones IoT confiables usando componentes de bajo costo. La mediana móvil resultó crítica para compensar las limitaciones del DHT11, reduciendo la desviación estándar de las lecturas en un 42% comparado con el promedio simple.

El suavizado exponencial (α=0.4) mostró un balance óptimo entre capacidad de respuesta y estabilidad, filtrando eficientemente las fluctuaciones térmicas de corto plazo sin introducir retardos significativos. La implementación del IQR con ventana móvil demostró ser 37% más efectiva que los umbrales fijos para detectar anomalías en ambientes dinámicos.