# EcoMonitor

Alumnos:
- Ulises Jaramillo Portilla — A01798380
- Jesús Ángel Guzmán Ortega — A01799257
- Sebastian Espinoza Farías — A01750311
- Santiago Villazón Ponce de León — A01746396
- Luis Ubaldo Balderas Sanchez — A01751150
- José Antonio Moreno Tahuilán — A01747922

## 1. Introducción

EcoMonitor es un **sistema de monitoreo ambiental completo** que integra hardware, software, aplicaciones móviles y análisis con **inteligencia artificial**.
Su objetivo es recolectar datos ambientales (temperatura, humedad, luz, calidad del aire, etc.), almacenarlos y visualizarlos en tiempo real, además de aplicar **modelos predictivos y detección de anomalías**.

Este proyecto fue desarrollado como parte de la **Práctica 3 de Dispositivos embebidos con sensores e interfaces móviles** del módulo *Integración de HW para ciencia de datos*.

---

# 2. Arquitectura General

El sistema está compuesto por tres grandes bloques:

1. **Hardware embebido (IoT)**
    - Arduino UNO con sensores:
        - DHT22 (temperatura y humedad).
        - LDR (intensidad lumínica).
        - Sensor MQ-135 (calidad del aire, en versiones extendidas).
    - Almacenamiento en CSV vía módulo SD o transmisión por puerto serie.
2. **Backend API (Flask)**

- API REST ligera en Python.
- Ingesta de datos desde **Google Sheets** o CSV local.
- Endpoints:
  - / (info general).
  - /data (datos con filtros por sensor, dispositivo y fechas).
  - /sensors (lista de sensores).
  - /devices (lista de dispositivos).
- Compatible con despliegue en **Docker**.
- Gestión de errores (404, 500, validación de fechas).

3. **Frontend móvil (Flutter)**
   - Aplicación multiplataforma (Android, iOS, Web).
   - Interfaz responsiva tipo **dashboard ambiental**.
   - Gráficas de series temporales y filtros por sensor/dispositivo.
   - Conexión directa al Backend API.
   - Soporta **hot reload** y builds de producción.
4. **Machine Learning & Análisis Predictivo**
   - Procesamiento de datos con **pandas/numpy**.
   - **Detección de anomalías** con *Isolation Forest*.
   - Modelos de predicción:
     - Regresión Lineal.
     - Random Forest.

        - LSTM (deep learning para series temporales, opcional si TensorFlow está disponible).
    - Predicciones de valores futuros (hasta 24 horas).
    - Visualizaciones:
        - Series temporales con anomalías.
        - Distribuciones.
        - Predicciones futuras.
        - Matrices de correlación.

---

# 3. Objetivos de la práctica

- Diseñar un sistema embebido de **monitoreo ambiental** con Arduino y sensores.
- Almacenar datos en **CSV** y transmitirlos vía puerto serie.
- Construir un **Backend API** que sirva como capa de acceso a los datos.
- Desarrollar un **Frontend móvil multiplataforma** para visualización de la información.
- Implementar un **modelo de Machine Learning** que permita:
    - Detectar anomalías en las mediciones.
    - Predecir valores futuros de sensores ambientales.

---

# 4. Desarrollo por componentes

## Hardware IoT (Arduino)

- Código en C++ usando librería **Adafruit DHT**.

- Lecturas de temperatura, humedad y luz cada 5 segundos.
- Envío de datos en formato **JSON** vía puerto serie.
- Ejemplo de salida:

```
JSON
{ "temperature": 25.3, "humidity": 43.7, "light": 320 }
```

## Backend (Flask API)

- API REST ligera con **Flask + pandas**.
- Carga datos desde Google Sheets (CSV export) o un archivo local.
- Ejemplo de consulta:

```
None
GET /data?sensor=temperature&device_id=esp32-1&start_date=2024-01-01
```

- Respuesta:

```
JSON
{
 "records": 123,
 "filters": {"sensor":"temperature","device_id":"esp32-1","start_date":"2024-01-01"},
 "data": [

{"timestamp":"2024-01-02T12:00:00Z","deviceId":"esp32-1","value":23.4,"unit":"°C","sensor":"
temperature","type":"numeric"}
 ]
}
```

## Frontend (Flutter)

- Proyecto en **Flutter/Dart**.
- Módulos principales:
    - lib/models/ → estructuras de datos de sensores.
    - lib/services/ → cliente API para conectarse al backend.
    - lib/presentation/ → pantallas y gráficos.
- **Funciones clave:**
    - Mostrar lista de sensores y dispositivos.
    - Gráficas interactivas de series temporales.
    - Adaptación multiplataforma (Android, iOS, Web).

### Inteligencia Artificial (Python ML)

- **Preprocesamiento de datos:** limpieza, conversión de fechas, encoding categórico.
- **Detección de anomalías:** Isolation Forest con ~5–10% de outliers.
- **Predicción:**
    - *Random Forest*: buena precisión en variables no lineales.
    - *Regresión Lineal*: base de comparación.
    - *LSTM*: aprendizaje profundo para series temporales.
- **Predicciones futuras:** hasta 24 horas, generando series simuladas.
- **Visualización:** gráficos automáticos exportados en PNG.

---

# 5. Resultados

- Se implementó un **sistema IoT funcional** que recolecta datos ambientales.

- El **Backend API** permite filtrar y consultar datos históricos en JSON.
- El **Frontend Flutter** brinda una interfaz amigable de acceso a la información.
- El **modelo de ML** detectó anomalías y generó predicciones confiables de temperatura, humedad y $CO_2$.
- Visualizaciones gráficas facilitaron la interpretación de resultados.

---

# 6. Conclusiones

- El proyecto integró **hardware, software, móvil y ML** en un sistema unificado.
- La arquitectura es **escalable y modular**, lo que permite añadir sensores o expandir la API fácilmente.
- El uso de **Machine Learning** aporta valor real al sistema, al detectar condiciones anómalas y anticipar tendencias.
- La app móvil (Flutter) demostró la importancia de la visualización y accesibilidad multiplataforma.
- EcoMonitor es una base sólida para aplicaciones **ambientales, agrícolas, industriales y educativas**.

---

# 7. Posibles mejoras

- Integrar almacenamiento en **bases de datos en la nube** (PostgreSQL, Firebase, InfluxDB).
- Desplegar la API en un servidor productivo (AWS, GCP, Heroku).
- Mejorar el modelo de predicción usando **Redes Neuronales Avanzadas** (Transformers para series temporales).
- Implementar un **dashboard web** adicional con Streamlit o Grafana.