# Práctica 2 – Comunicación al Fin del Mundo: MQTT & Meshtastic

Asignatura: Programación Orientada a Objetos

Curso 2025


## Práctica 2: Extensión de la Práctica 1

En esta segunda práctica, el objetivo principal será la **ampliación de la Práctica 1**. Esto implica que, si la Práctica 1 no funciona correctamente, esta nueva práctica tampoco podrá hacerlo, ya que depende directamente de su correcto funcionamiento. Por tanto, antes de comenzar, será necesario **revisar, implementar o corregir** la Práctica 1 en caso de ser necesario.

El propósito de esta práctica es que el estudiante adquiera experiencia en el **diseño y desarrollo de aplicaciones** siguiendo el paradigma de la **Programación Orientada a Objetos (POO)**, aplicadas a contextos prácticos y reales.

A partir de las clases desarrolladas en la Práctica 1, el estudiante deberá **ampliarlas** incorporando las siguientes características vistas en clase:

- **Clases abstractas:** al menos una clase abstracta debe ser implementada.
- **Herencia múltiple:** se debe incluir como mínimo un caso de herencia múltiple.
- **Programación defensiva:** incluir el uso de `try`, `except`, `raise`, entre otros mecanismos de manejo de errores.

- **Decoradores:** aplicar al menos un decorador (por ejemplo, utilizando el símbolo `@`).

## Criterios de evaluación

- **Funcionamiento general del programa (5 puntos):** El sistema deberá ejecutar correctamente todas las funcionalidades solicitadas sin presentar errores durante su ejecución. Se valorará la estabilidad, el cumplimiento de los requisitos y la correcta interacción entre los distintos módulos del código. En caso contrario, se aplicará una penalización de **hasta 5 puntos**.
- **Implementación de conceptos avanzados de POO (3 puntos):** Se evaluará la correcta aplicación de los siguientes elementos:
  - Uso de al menos una **clase abstracta**.
  - Implementación de **herencia múltiple**.
  - Aplicación de **programación defensiva** (`try`, `except`, `raise`, etc.).
  - Uso de **decoradores** mediante el símbolo `@`.
  - Aplicación de conceptos de **genericidad**.
  - Excepciones personalizadas.
- **Calidad del diseño y del código (1 punto):** Se valorará la legibilidad, la estructura modular, la claridad de los nombres de variables y funciones, la presencia de comentarios adecuados y el uso de buenas prácticas de programación.
- **Informe técnico (0.5 puntos):** El informe deberá incluir los siguientes apartados obligatorios:
  - Portada
  - Introducción
  - Objetivos
  - Desarrollo (incluyendo capturas de pantalla y explicación del código)
  - Conclusiones
  - Anexos (librerías y dependencias utilizadas)

  **Nota:** La ausencia de cualquiera de estos apartados supondrá una penalización de **1 punto**.
- **Repositorio Git (0.5 puntos):** El repositorio deberá estar correctamente organizado y contener un archivo `README.md` con una descripción clara del proyecto, instrucciones de ejecución y dependencias. Un repositorio sin `README` o mal estructurado tendrá una penalización de **2 puntos**.
- **Extras documentados (hasta +1 punto adicional):** Los extras implementados por los estudiantes y correctamente documentados en la memoria serán tenidos en cuenta de forma positiva.

## Extras

Cada extra podrá ser desarrollado de manera opcional por los estudiantes y será **catalogado según su nivel de dificultad**. Si algún estudiante desea realizar un extra clasificado con **tres estrellas (★ ★ ★)**, deberá comunicarlo previamente al profesor para las consideraciones correspondientes.

| Nivel | Descripción del Extra | Dificultad | Valor |
|---|---|---|---|
| 1 | Crear una interfaz gráfica básica. | ★ | 0.25 puntos (3er examen práctico) |
| 2 | Interfaz gráfica con visualización de coordenadas en un mapa. | ★★ | 0.5 puntos (3er examen práctico) |
| 3 | Interfaz gráfica con mapa y comunicación con un dron. | ★★★ | 1 punto (3er examen práctico) |
| 4 | Interfaz gráfica con mapa y comunicación con un robot. | ★★★ | 1 punto (3er examen práctico) |

**Nota:** Los dos últimos extras (niveles 3 y 4), si son realizados y documentados correctamente, tendrán una consideración especial en la evaluación.

# Recursos y enlaces necesarios

## Interfaz Gráfica de Usuario

- Interfaz Gráfica en Python (Tkinter)
- TkinterMapView

## MQTT

- Eclipse Mosquitto (broker MQTT)
- Paho-MQTT (cliente en Python)
- Documentación oficial Eclipse Paho
- MQTT Explorer (cliente gráfico)

## Meshtastic

- Página oficial Meshtastic
- Repositorio GitHub oficial
- Librería Python Meshtastic
- Documentación API Meshtastic
- Ejemplos en Python

## Python y POO

- Tutorial oficial de Python (clases y POO)
- Guía práctica (Real Python)

## Ejemplos extra

- Ejemplo Paho-MQTT (publicador/suscriptor)
- Ejemplo Meshtastic en Python

## Fecha de entrega

La práctica deberá ser entregada como máximo el **02 de diciembre de 2025**.