

Práctica 1 – Comunicación al Fin del Mundo: MQTT & Meshtastic

Asignatura: Programación Orientada a Objetos

Curso 2025



La humanidad enfrenta su último desafío. Una tormenta solar ha arrasado con gran parte de las infraestructuras de comunicación global. Las redes móviles han colapsado, los satélites están inactivos, y sólo quedan pequeños nodos de dispositivos capaces de transmitir mensajes críticos a través de protocolos resilientes como **MQTT** y la red **Meshtastic**.

Tu misión: diseñar y programar un sistema de comunicación autónomo que garantice la supervivencia de las últimas comunidades humanas.

Objetivo de la práctica

El objetivo es que el estudiante adquiera experiencia en el diseño y desarrollo de aplicaciones bajo el paradigma de la **Programación Orientada a Objetos (POO)**, aplicadas a la comunicación entre dispositivos.

En esta práctica inicial, se utilizarán los protocolos **MQTT** y **Meshtastic** para el **envío y recepción de mensajes desde la terminal de comandos**.

Nota: En futuras prácticas se ampliará el sistema para capturar y transmitir datos de sensores en tiempo real, que podrán ser graficados con Matplotlib. En esta versión introductoria nos centraremos únicamente



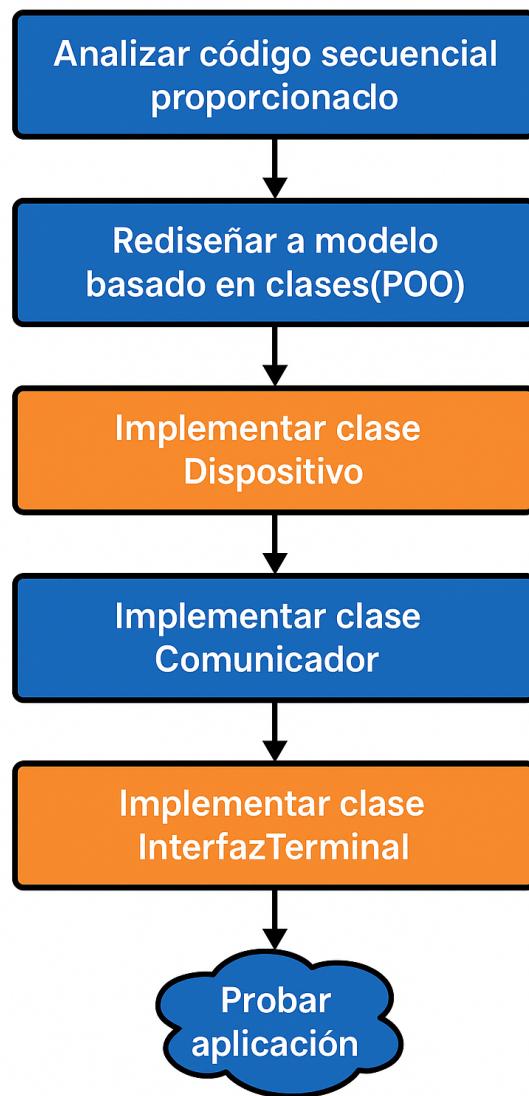
Tareas a realizar

Se pondrá a disposición del estudiante un código base en Python, desarrollado de manera **secuencial**. Dentro de este repositorio es especialmente importante revisar los ejemplos contenidos en la carpeta MQTT, ya que servirán como punto de partida para el rediseño orientado a objetos que se requiere en esta práctica.

El estudiante deberá:

1. Analizar el código proporcionado.
2. Rediseñarlo a un modelo basado en clases (POO).
3. Implementar una clase **Dispositivo** (atributos: ID, protocolo, estado de conexión, historial de mensajes y posiciones GPS).
4. Implementar una clase **Comunicador** con métodos para enviar y recibir mensajes (MQTT y Meshtastic).
5. Implementar una clase **InterfazTerminal** que permita la ejecución desde línea de comandos.
6. Asegurar que los mensajes y las coordenadas GPS recibidas queden almacenados de forma persistente (por ejemplo en un archivo .txt o .json).
7. Utilizar la conexión **MQTT vista en clase** para el almacenamiento de datos de sensores en formato **JSON**.
8. Incluir un **menú de selección** para que el usuario pueda elegir si quiere recibir datos de la red Meshtastic o los datos de los sensores por MQTT.
9. Junto con el código fuente deberá entregarse un archivo de texto denominado **REQUIREMENTS.txt**, en el que se especifiquen todas las librerías utilizadas en el desarrollo de la práctica. La ausencia de este archivo supondrá una penalización de **0.5 puntos** en la nota final.
10. Probar la aplicación mediante ejemplos como:

```
python supervivencia.py --modo mqtt --enviar "SOS: Quedan provisiones en el refugio"  
python supervivencia.py --modo meshtastic --recibir
```



Parámetros de conexión de ejemplo

Para facilitar la conexión a la red, se proporciona un conjunto de parámetros de configuración que los estudiantes pueden utilizar como punto de partida en sus programas:

```
"mqtt": {  
    "broker": "mqtt.meshtastic.org",  
    "port": 1883,  
    "username": "meshdev",  
    "password": "large4cats",  
    "root_topic": "msh/EU_868/2/e/",  
    "channel": "TestMQTT"  
},  
"encryption": {
```



```
"key_128": "6bg+mfRpLkXome6druAYCg==",
"key": "ymACgCy9Tdb8jHbLxUxZ/4ADX+BWLOGVihmKHcHTVyo=",
"key_default": "AQ=="
}
```

Grado en Tecnologías Digitales para la Empresa

Nota: Estos valores son de uso educativo. Los estudiantes pueden probar sus conexiones con ellos, pero también deberán ser capaces de modificarlos si se requiere un despliegue en otro broker o con claves diferentes.

Resultados esperados

El estudiante deberá entregar:

- Código fuente en Python, documentado y organizado en clases, en Git con un README.
- Almacenamiento persistente de los mensajes, posiciones GPS y datos de sensores recibidos en formato JSON.
- Un informe en PDF con:
 - Diseño de clases y diagrama UML.
 - Explicación del flujo de mensajes entre dispositivos.
 - Ejemplos de uso del sistema (capturas de terminal).
 - Evidencia de los archivos generados con historial de mensajes, posiciones GPS y datos de sensores.

Criterios de evaluación

- Transformación correcta del código secuencial a POO.
- Funcionamiento del envío/recepción de mensajes por MQTT y Meshtastic.
- Calidad del código (legibilidad, documentación, buenas prácticas).
- Calidad y claridad del informe entregado.
- El informe deberá contener de manera obligatoria los siguientes apartados:
 - Introducción
 - Objetivos
 - Desarrollo (con capturas de pantalla)
 - Conclusiones
 - Anexos (librerías utilizadas)

Nota: La ausencia de cualquiera de estos apartados supondrá una penalización de **1 punto** en la nota final de la práctica.

Recursos y enlaces necesarios

MQTT

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

Meshtastic

Grado en Tecnologías Digitales para la Empresa

- 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 **20 de octubre de 2025**.

