Laboratorio 7 Redes
IoT: Estación Meteorológica

Andy Fuentes – 22944, Davis Roldan –22672

16 de noviembre de 2025

1. Objetivos

- Implementar y simular una solución típica de IoT utilizando sensores y un broker de mensajes.
- Utilizar herramientas de software para aplicaciones de IoT y Edge Computing (Apache Kafka).
- Resolver los requerimientos del laboratorio bajo restricciones de formato y tamaño de *payload*.

2. Simulación de sensores y envío (3.1 y 3.2)

Se implementó una función `generar_medicion()` que simula:

- Temperatura en $[0, 110]$ °C con distribución gaussiana (media cercana a 25° y desviación moderada), recortando el valor al rango permitido y redondeando a dos decimales.
- Humedad relativa entera en $[0, 100]$ %.
- Dirección del viento en {`N, NO, O, SO, S, SE, E, NE`}.

Cada lectura se empaqueta en JSON:

```
{"temperatura": 23.45, "humedad": 53, "direccion_viento": "SO"}
```

Un Kafka Producer se conecta al broker `iot.redesuvg.cloud:9092` y publica las mediciones en el topic `22944` cada 15–30 segundos de forma continua.

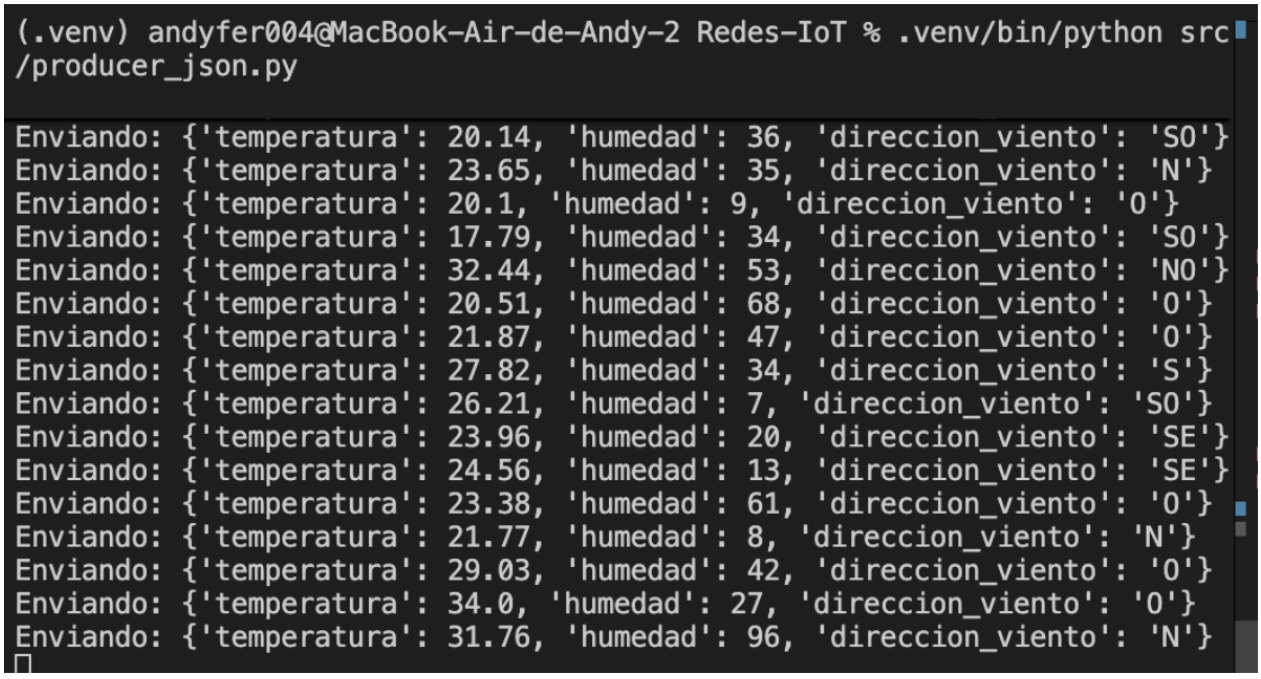

Figura 1: Producer JSON enviando mediciones simuladas al topic `22944`.

Preguntas 3.1

¿A qué capa pertenece JSON/SOAP según el Modelo OSI y por qué?
JSON y SOAP se ubican en la **capa de aplicación** del modelo OSI, porque definen el formato de los datos que intercambian las aplicaciones, no el transporte ni el enlace. Normalmente viajan sobre protocolos como HTTP/TCP.

¿Qué beneficios tiene utilizar un formato como JSON/SOAP?
Permiten interoperabilidad entre lenguajes y plataformas, los mensajes son legibles y fáciles de depurar, y ayudan a desacoplar Producer y Consumer: mientras ambos entiendan el formato, la implementación interna puede cambiar sin romper la comunicación.

3. Consumo y despliegue de datos (3.3)

Se implementó un Kafka Consumer suscrito al topic `22944` que:

1. Deserializa el JSON recibido.
2. Extrae temperatura y humedad.
3. Acumula los valores en listas.
4. Actualiza una gráfica con `matplotlib` en cada mensaje.

```
(.venv) andyfer004@MacBook-Air-de-Andy-2 Redes-IoT % .venv/bin/python src
/consumer_json.py
```

Figura 2: Ejecución del consumer JSON leyendo del topic 22944.

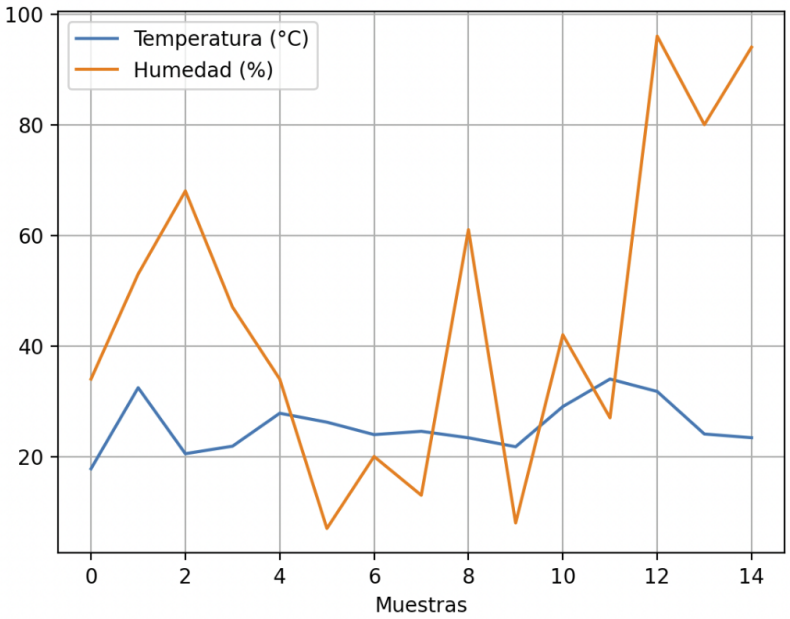

Figura 3: Gráfica en tiempo casi real de temperatura y humedad usando mensajes JSON.

Preguntas 3.3

¿Ventajas y desventajas de un esquema Pub/Sub con Kafka?
Como ventajas, el modelo Pub/Sub desacopla productores y consumidores, escala bien con muchos clientes, permite procesar eventos casi en tiempo real y ofrece tolerancia a fallos. Como desventajas, añade complejidad de infraestructura, requiere monitoreo y configuración cuidadosa, y puede ser más difícil de depurar que un esquema directo petición–respuesta.

¿Para qué aplicaciones tiene sentido usar Kafka? ¿Para cuáles no?
Es muy útil en sistemas con alto volumen de eventos o logs: IoT, monitoreo, analítica en tiempo real, microservicios basados en eventos, etc. No se justifica tanto en aplicaciones pequeñas o con pocas peticiones, donde una API REST simple o una base de datos ya resuelven el problema sin montar un clúster Kafka.

4. Payload restringido a 3 bytes (3.4)

Para modelar redes IoT con *payload* máximo de 3 bytes (24 bits) se diseñó un formato compacto:

```
[ temp(14 bits) | humedad(7 bits) | viento(3 bits) ]
```

- La temperatura se escala a centésimas: `temp_int = round(temp * 100)` con rango $[0, 11000]$. Como $2^{14} = 16384$, cabe en 14 bits.
- La humedad, entera en $[0, 100]$, cabe en 7 bits.
- La dirección del viento se codifica como un índice de 3 bits (0–7).

Se implementaron dos funciones:

- `encode_payload(temp, hum, dir)`: empaqueta los 24 bits y devuelve 3 bytes.
- `decode_payload(bytes)`: recupera temperatura, humedad y dirección desde esos 3 bytes.

El producer compacto envía los bytes al topic `22944-compacto`, como se ve en la Figura 4.

```
(.venv) andyfer004@MacBook-Air-de-Andy-2 Redes-IoT % .venv/bin/python src/producer_compacto.py
Producer compacto corriendo... Ctrl+C para salir
Enviando (compacto): 30.188229272632018 77 N -> b'/.h'
Enviando (compacto): 35.43351333322938 37 SO -> b'7]+'
Enviando (compacto): 30.743642101450895 28 NE -> b'0\x08\xe7'
Enviando (compacto): 22.135290858167988 50 SE -> b'"\x99\x95'
Enviando (compacto): 26.086240277931648 32 SO -> b'(\xc5\x03'
Enviando (compacto): 25.84374294212407 30 S -> b'(`\xf4'
Enviando (compacto): 32.00432108897834 93 SE -> b'2\x02\xed'
Enviando (compacto): 23.401343706781763 46 N -> b'$\x91p'
Enviando (compacto): 20.001158410852938 63 SO -> b'\x1fA\xfb'
```

Figura 4: Producer compacto: valores de sensores y representación en 3 bytes.

El consumer compacto se suscribe al mismo topic, decodifica cada mensaje y vuelve a graficar temperatura y humedad (Figura 5).

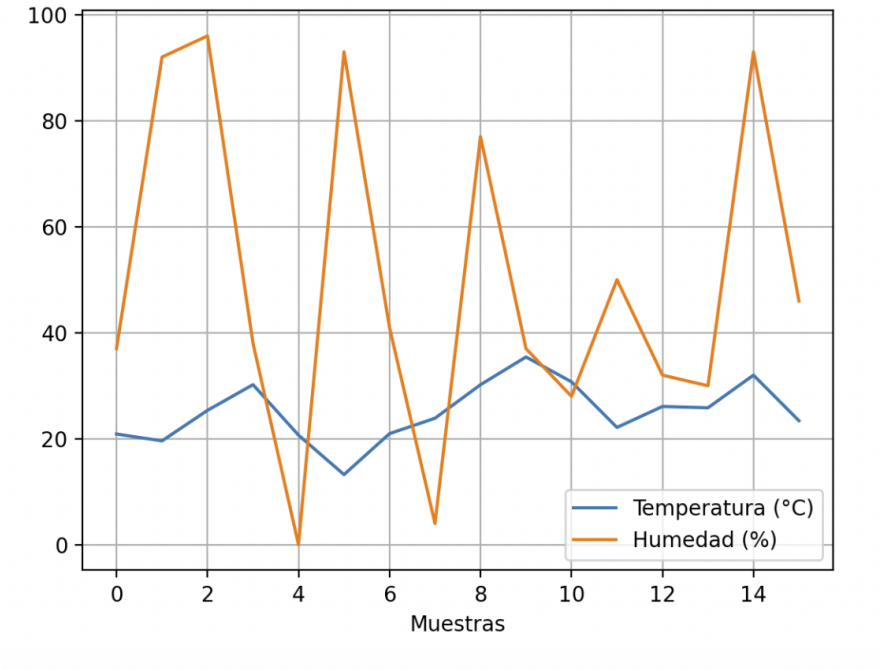

Figura 5: Gráfica de temperatura y humedad usando el payload compacto.

Preguntas 3.4

¿Qué complejidades introduce el tener un payload pequeño?
Obliga a diseñar cuidadosamente el empaquetado de bits, cuantizar y perder precisión en algunos campos, y coordinar muy bien el encoder y el decoder. Un solo bit mal colocado puede afectar varios datos.

¿Cómo hacer que la temperatura quepa en 14 bits?
Se limita la temperatura al rango $[0, 110]$ y se multiplica por 100 para trabajar con centésimas: máximo 11000. Ese valor cabe en 14 bits porque $2^{14} = 16384$. Al decodificar, se divide entre 100 para obtener de nuevo los grados Celsius.

¿Qué pasaría si la humedad también fuera float con un decimal?
Habría que escalarla (por ejemplo `hum_int = round(hum * 10)`), lo que exige más bits. Con solo 24 bits totales habría que sacrificar precisión en temperatura o humedad, reducir el rango o dejar de enviar alguna variable; en otras palabras, priorizar qué dato necesita más precisión.

¿Qué parámetros o herramientas de Kafka ayudan si las restricciones son más fuertes?
Se puede aprovechar la compresión de mensajes (gzip, snappy), el *batching* de mensajes y parámetros como `batch.size` y `linger.ms` para reducir overhead. Sin embargo, el límite de 3 bytes viene más de la red IoT que de Kafka, así que el broker ayuda más a optimizar cuántos mensajes se envían y cómo se agrupan.

5. Conclusiones

En este laboratorio se integraron conceptos de IoT, Edge Computing y sistemas de mensajería. Implementar la estación meteorológica sobre Kafka permitió ver en la práctica las ventajas del modelo Pub/Sub para procesar flujos de eventos, y el ejercicio de compactar los mensajes a 3 bytes obligó a razonar en términos de bits, rangos y precisión, como ocurre en redes IoT reales con restricciones fuertes.