Universidad del Valle de
Guatemala
Redes
Javier Prado - 21486

Laboratorio 7
El internet de las cosas

3.1 Simulación de un Sensor

Crear 3 sensores en sensorSimulator.py

Temperatura: float en [0, 110.00] °C con 2 decimales.

Humedad: int en [0, 100] %.

Dirección del viento: escoge aleatoriamente la dirección del viento {N, NO, O, SO, S, SE, E, NE}

```
Lectura #1:
  Temperatura: 22.72°C
  Humedad: 32%
  Dirección del viento: NE
  Timestamp: 2025-11-16T23:58:59.155921

  JSON: {
  "temperatura": 22.72,
  "humedad": 32,
  "direccion_viento": "NE",
  "timestamp": "2025-11-16T23:58:59.155921"
}

Lectura #2:
  Temperatura: 24.73°C
  Humedad: 56%
  Dirección del viento: SO
  Timestamp: 2025-11-16T23:58:59.157078

  JSON: {
  "temperatura": 24.73,
  "humedad": 56,
  "direccion_viento": "SO",
```

Universidad del Valle de
Guatemala
Redes
Javier Prado - 21486

```
  "humedad": 73,
  "direccion_viento": "O",
  "timestamp": "2025-11-17T00:04:20.745975"
}

Lectura #4:
  Temperatura: 12.07°C Humedad: 75%
  Humedad: 75%
  Dirección del viento: NO
  Timestamp: 2025-11-17T00:04:20.746478

  JSON: {
  "temperatura": 12.07,
  "humedad": 75,
  "direccion_viento": "NO",
  "timestamp": "2025-11-17T00:04:20.746478"
}

Lectura #5:
  Temperatura: 28.24°C Humedad: 72%
  Humedad: 72%
  Dirección del viento: O
  Timestamp: 2025-11-17T00:04:20.746877
```

Responda: ¿A qué capa pertenece JSON/SOAP según el Modelo OSI y por qué?

UN JSON/SOAP pertenecen a la capa 7 la capa de aplicación. Esto debido a que ambos definen cómo las aplicaciones estructuran y representan los dataos que se intercambian. En estos se ve el formato del mensaje como los campos, tipos, estructura. Pero no se encargan de cómo se envían físicamente los bits por la red.

Responda: ¿Qué beneficios tiene utilizar un formato como JSON/SOAP?

Los beneficios de usar un formato como JSON/SOAP son los siguientes:

- Distintos lenguajes pueden entender el mensaje al ser ampliamente estándar.
- Son legibles para el ojo humano, por lo que facilita revisar el mensaje en crudo.
- Permite representar los datos de una forma estructurada (objetos, listas, campos con nombre)
- Es fácil de agregar nuevos campos sin romper necesariamente a los clientes existentes.

3.2 Envío de Datos al Server Edge

En kafkaProducer.py se enviará la data cada 15 segundos con el Topic 21486 al sever iot.redesuvg.cloud:9092 correspondiente a la IP: 147.182.219.133. Se utiliza el puerto estándar de Kafka (9092).

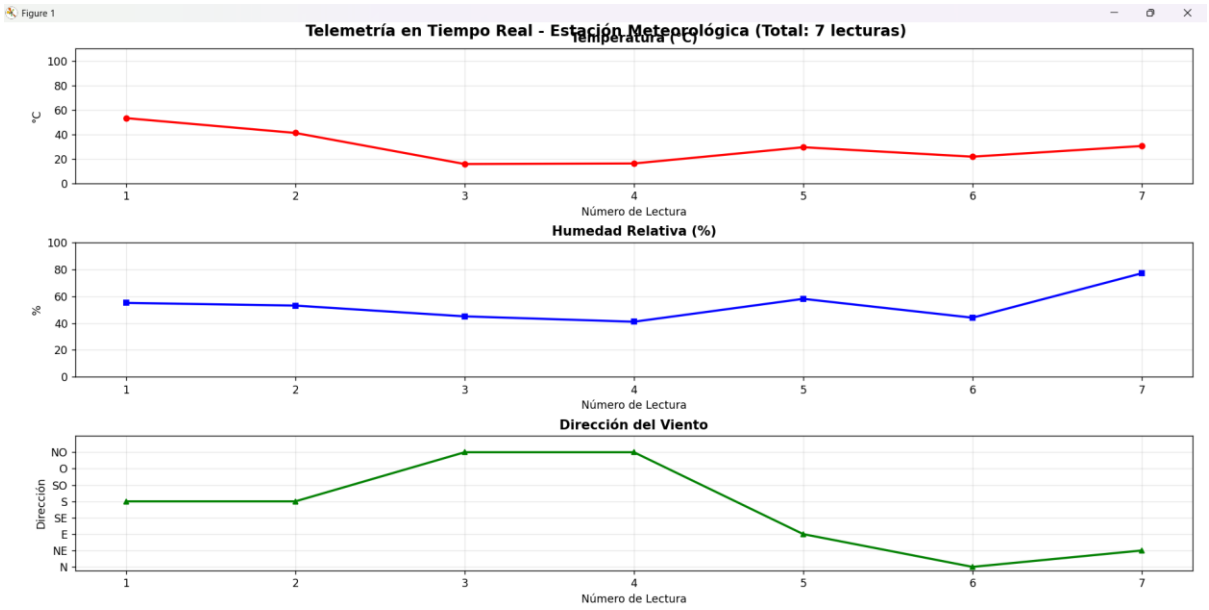
```
--- Lectura #1 ---

[00:25:04] Mensaje enviado exitosamente
  Topic: 21486
  Partición: 0
  Offset: 1
  Datos: {
  "temperatura": 6.46,
  "humedad": 47,
  "direccion_viento": "S"
}
Esperando 15 segundos para la proxima lectura...
```

3.3 Consumir y Desplegar Datos Meteorológicos

El kafkaConsumer.py implementado esta suscrito al topic 21486 para escuchar sus mensajes entrantes

Universidad del Valle de
Guatemala
Redes
Javier Prado - 21486



Universidad del Valle de
Guatemala
Redes
Javier Prado - 21486

```
[01:01:35] Mensaje enviado exitosamente
  Topic: 21486
  Partición: 1
  Offset: 1
  Datos: {
  "temperatura": 15.82,
  "humedad": 45,
  "direccion_viento": "NO"
}
Esperando 15 segundos para la proxima lectura...

--- Lectura #6 ---

[01:01:50] Mensaje enviado exitosamente
  Topic: 21486
  Partición: 0
  Offset: 4
  Datos: {
  "temperatura": 21.81,
  "humedad": 44,
  "direccion_viento": "N"
}
Esperando 15 segundos para la proxima lectura...

--- Lectura #7 ---

[01:02:05] Mensaje enviado exitosamente
  Topic: 21486
  Partición: 0
  Offset: 5
  Datos: {
  "temperatura": 30.54,
  "humedad": 77,
  "direccion_viento": "NE"
}
```

Responda: ¿Qué ventajas y desventajas considera que tiene este acercamiento basado en Pub/Sub de Kafka?

Kafka permite desacoplar fuertemente a productores y consumidores: el productor solo publica eventos en un topic y no necesita conocer quién los consume. Esto facilita escalar el sistema (se pueden agregar más consumers sin cambiar a los producers), mejora la tolerancia a fallos gracias a la replicación y permite procesar flujos de datos en tiempo real. Además, la persistencia de eventos en el log de Kafka permite reprocesar histórico, algo muy útil para análisis y depuración. A cambio, Kafka introduce más complejidad de infraestructura y operación que un simple esquema petición–respuesta. Hay que administrar brokers, topics, particiones y grupos de consumidores. También puede añadir algo de latencia extra y overhead cuando los

volúmenes de datos son muy pequeños o el caso de uso es muy simple, donde un enfoque directo HTTP/REST sería suficiente.

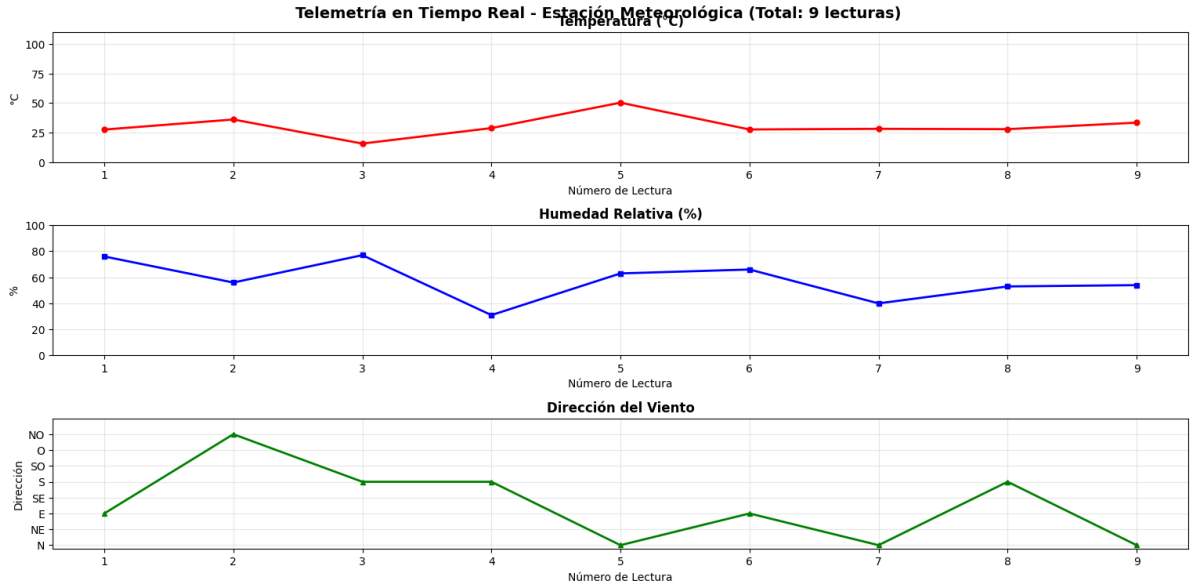
Responda: ¿Para qué aplicaciones tiene sentido usar Kafka? ¿Para cuáles no?

Tiene sentido usar Kafka en aplicaciones que generan o consumen grandes volúmenes de datos en flujo: telemetría de IoT, monitoreo de sistemas, logs de aplicaciones, procesamiento de eventos en tiempo real, integración entre microservicios y pipelines de analítica/Big Data. No es la mejor opción para operaciones aisladas y sin flujo continuo (por ejemplo, un formulario sencillo web que guarda un registro en una base de datos), para transferir archivos grandes de forma puntual o como reemplazo de una base de datos transaccional tradicional.

3.4 IoT en Entornos con Restricciones

Se modifico para que el producer y consumer pudieran codificar y decodificar los datos en 3 bytes.
Para la temperatura se usaron 14 bits en el cual se convirtio en centésimas [0-11000]
Para la humedad se usarono 7 bits valor entero [0-100]
para la direccion del viento 3 bits, el mapeo es el siguiente N=0, NO=1, O=2, SO=3, S=4, SE=5, E=6, NE=7.
Con estos cambios en la data se envian 3 bytes de data por mensaje

Universidad del Valle de
Guatemala
Redes
Javier Prado - 21486




```
--- Lectura #8 ---

[01:52:54] Mensaje enviado exitosamente
  Topic: 21486
  Partición: 2
  Offset: 15
  Payload: 3 bytes = 24 bits
  Hex: 2bbdac
  Datos originales:
    Temperatura: 27.99°C
    Humedad: 53%
    Dirección: S
  Datos codificados:
    Temp int: 2799 (14 bits: 00101011101111)
    Hum int: 53 (7 bits: 0110101)
    Wind code: 4 (3 bits: 100)
    Packed 24 bits: 001010111011110110101100
  Datos decodificados (verificación):
    Temperatura: 27.99°C
    Humedad: 53%
    Dirección: S
Esperando 15 segundos para la proxima lectura...
```

Universidad del Valle de
Guatemala
Redes
Javier Prado - 21486

```
--- Lectura #9 ---

[01:53:09] Mensaje enviado exitosamente
  Topic: 21486
  Partición: 0
  Offset: 13
  Payload: 3 bytes = 24 bits
  Hex: 3459b0
  Datos originales:
    Temperatura: 33.5°C
    Humedad: 54%
    Dirección: N
  Datos codificados:
    Temp int: 3350 (14 bits: 00110100010110)
    Hum int: 54 (7 bits: 0110110)
    Wind code: 0 (3 bits: 000)
    Packed 24 bits: 001101000101100110110000
  Datos decodificados (verificación):
    Temperatura: 33.5°C
    Humedad: 54%
    Dirección: N
Esperando 15 segundos para la proxima lectura...
```

Responda: ¿Qué complejidades introduce el tener un payload restringido (pequeño)?

El diseño de codificacion no basta con mandar un JSON dado que hay que diseñar la reparticion de cuantos bits se enviaran para cada campo, por lo que implica más trabajo y posibilidades de errores. Puede que exista perdida y que el mensaje ya no sea legible para los humanos, porque envez de poder leer {"temperatura": 25.37, "humedad": 72}, se ve algo como 1001010001.., Por lo que al tener pocos bits muchas veces hay que reducir precisión o acotar rangos, si no se diseña bien puede que ciertos valores no se puedan representar correctamente.

Responda: ¿Cómo podemos hacer que el valor de temperatura quepa en 14 bits?

En vez de guardarlo como float se guarda como un entero. El rango de temperatura es de 0° a 110° C . Se escoge una precisión razonable para representar la temperatura como se hizo en este caso se usaron centésimas de grado. Por lo que se convierte la temperatura tipo float a int, para luego si es 25-37 --> 2537 es el valor , el valor máximo es 100.00°c equivalente a 11000. Por lo que se verifica que quepa en 14 bits 2^14= 16383 y el valor es 11000 menor a 16383 por lo que cabe en 14 bits.

Universidad del Valle de
Guatemala
Redes
Javier Prado - 21486

Responda: ¿Qué sucedería si ahora la humedad también es tipo float con un decimal? ¿Qué decisiones tendríamos que tomar en ese caso?

En este caso tenemos complicaciones dado que si la humedad pasa de 0-100 a float con decimal 10.2% ya no canen en los 0-100 con 7 bits de forma directa. Para representar la humedad hasta 1000 necesitamos 10 bits para representar hasta 1000, por lo que reduciria la presicion en la humedad. Por lo que habria que sacrificar la precisión, rango o tamaño de mensaje , o rediseñar el protocolo.

Responda: ¿Qué parámetros o herramientas de Kafka podrían ayudarnos si las restricciones fueran aún más fuertes?

Podria usar compression.type que , si empiezo a enviar más campos la compresión reduce el tamaño total de datos que viajan por la red y se guardan en kafka. Con batch.size, linger.ms agrupan varios mensajes en un batch antes de enviarlos para pro ejemplo con linger.ms permite esperar unos milisegundos para acumular más mensajes en el mismo paquete .