Arquitectura y Conectividad

Cuestionario N°4

Profesor Ing. Morales Jorge Elias | https://github.com/JorEl057

Miembros:

- Durigutti, Vittorio | GitHub: https://github.com/vittoriodurigutti
- Zalazar, Joaquín | GitHub: https://github.com/breaakerr
- Marquez, José | Github: https://github.com/marquezjose
- Lujan, Luciano | Github: https://github.com/lucianoilujan
- Velez, Nahuel | Github: https://github.com/Lucasmurua19
- Juncos, Lisandro | Github: https://github.com/Lisandro-05
- Garzón, Joaquín | Github: https://github.com/Joacogarzonn
- Guzmán, Maria |Github: https://github.com/lilenguzman01

Actividad: (Click en las preguntas para link a las respuestas)

1) ¿Qué es una Comunicación REST?, ¿Para qué se usan? Ejemplifique.

2) ¿Qué es un Formato de datos JSON?, ¿Para qué se usan? Ejemplifique.

3) Implementar un pequeño código JSON, donde un cliente solicita una pizza.(por lo menos 4 o 5 tipos de pizza) (2 tamaños) (precios).

4) Implementar un código JSON, para comunicar un sensor de temperatura y humedad con un ESP32, Arduino, simulando los mismos en WOKWI, Proteus, LabView; etc. ¿Cuáles serían los campos mínimos para hacer la implementación?

Fecha de presentación: 23/04/25.

1- ¿Qué es una Comunicación REST?, ¿Para qué se usan? Ejemplifique.

Comunicación **REST** (Representational State Transfer)

REST es un forma arquitectónica que define cómo interactúan los sistemas en una red, usa principalmente el protocolo HTTP. Este sistemas RESTful intercambian datos en formato sencillo (como JSON o XML) mediante operaciones como GET, POST, PUT, DELETE, como vimos en la Actividad 3 . Se usa en aplicaciones web y sistemas IoT.

¿Para qué se usa?

REST se usa para diseñar servicios web que permiten:
Acceso y manipulación de recursos en un servidor.
Comunicación entre sistemas diferentes de manera ligera y eficiente.
Creación de APIs RESTful, las cuales permiten que aplicaciones o dispositivos se comuniquen con un servidor para consultar o modificar datos.

Ejemplo:

Vamos a suponer que una aplicación utiliza un API RESTful para mostrar datos de usuarios.
Solicitud REST (usando HTTP GET): *GET https://api.example.com/usuarios/123*
Respuesta del servidor (en formato JSON): *{ "id": 123, "nombre": "Luciano", "email": "luciano@example.com" }*
En este caso el cliente solicita información sobre un usuario con el ID 123 y el servidor responde con los datos en JSON.

2- ¿Qué es un Formato de datos JASON?, ¿Para qué se usan? Ejemplifique.

¿Qué es el formato JSON?

JSON (JavaScript Object Notation) es un formato ligero de intercambio de datos, basado en texto, fácil de leer y escribir tanto para humanos como para máquinas. Se originó en el lenguaje JavaScript, pero hoy es compatible con prácticamente todos los lenguajes de programación.

¿Para qué se utiliza?

JSON se utiliza principalmente para estructurar y transportar datos entre diferentes sistemas. Es muy común en entornos de desarrollo web, móviles, IoT, bases de datos y APIs modernas.

Estructura de un archivo JSON

Los datos en JSON se organizan como pares clave-valor, encerrados entre llaves {}.
Cada clave (key) debe ser un string entre comillas dobles y debe tener un valor asociado:

```
{
  "clave": "valor"
}
```

Los valores pueden ser:

- Cadenas de texto ("texto")
- Números (25, 3.14)
- Booleanos (true, false)
- Arreglos (listas) ([1, 2, 3])
- Objetos (diccionarios anidados)
- null (para valores vacíos)

Ejemplos de uso

En una API RESTful:

El cliente realiza una solicitud al servidor para obtener información sobre un usuario.
Solicitud GET:
GET /api/usuarios/1
Respuesta JSON:

```
{
  "id": 1,
  "nombre": "Ana",
  "email": "ana@example.com"
}
```

En un dispositivo IoT:

Un sensor de temperatura envía sus datos a un servidor central.

```
{
  "sensor_id": "DHT11",
  "temperatura": 25.6,
  "unidad": "Celsius",
  "ubicacion": "Cocina"
}
```

Este mensaje puede ser procesado por una plataforma que almacena, analiza y muestra los datos en tiempo real.

En almacenamiento local de datos estructurados:

Una aplicación puede guardar configuraciones de usuario en un archivo JSON.

```
{
  "tema": "oscuro",
  "idioma": "es",
  "notificaciones": true
}
```

Relación entre JSON y REST

En arquitecturas RESTful, JSON es el formato más utilizado para el intercambio de datos. La comunicación suele seguir el siguiente esquema:

- Cliente ➡️ Servidor (Request): El cliente envía datos en formato JSON mediante métodos HTTP como POST, PUT o PATCH.
- Servidor ➡️ Cliente (Response): El servidor responde con datos estructurados en JSON.

Esto permite una interacción clara, predecible y sencilla entre sistemas, sin importar el lenguaje o plataforma que utilicen.

Ventajas del formato JSON

- Legible para humanos
- Ligero (menor tamaño que XML, por ejemplo)
- Compatible con la mayoría de lenguajes de programación
- Ideal para sistemas distribuidos y en tiempo real

3- Implementar un pequeño código JSON, donde un cliente solicita una pizza. (por lo menos 4 o 5 tipos de pizza) (2 tamaños) (precios).

Se crea el documento formato json "pizzas.json" en el directorio "/C prototipo/Actividad N°4/.." ["./C prototipo/Actividad N°4"]

En el documento definimos el objeto pizza, con 3 características: nombre, tamaño, ingredientes.

```
{
"pizzas": [
   {
   "nombre": "",
   "tamaños": { },
   "ingredientes": { }
   },
```

```
]
}
```

nombre: un string, solo será el nombre de la variedad de pizza.

tamaño es un diccionario con el formato "tamaño":"cantidad de porciones"

```
"tamaños": {
  "chica": 4,
  "grande": 8
},
```

Los “ingredientes” se compone de 3 componentes. "salsa" y "aceitunas" que funcionaran como booleanos según tenga o no. y "topings" con la lista de posibles ingredientes.

```
"ingredientes": {
  "salsa": true/false,
  "aceitunas": true/false,
  "topings": [ ]
}
```

4- Implementar un código JSON, para comunicar un sensor de temperatura y humedad con un ESP32, Arduino, simulando los mismos en WOKWI.

¿Cuáles serían los campos mínimos para hacer la implementación?

```
#include <ArduinoJson.h>
```

Esta línea incluye la biblioteca ArduinoJson. ArduinoJson es una librería para trabajar con el formato JSON (JavaScript Object Notation) en placas como Arduino, ESP32, ESP8266, etc.

StaticJsonDocument<128> doc;

Se crea un objeto llamado doc de tipo StaticJsonDocument, el cual reserva 128 bytes de memoria estática para almacenar los datos en formato JSON. Este objeto servirá como contenedor para construir y organizar la estructura del mensaje JSON.

doc["Temperatura"] = temperatura;

Asigna un valor a la clave "Temperatura" dentro del objeto JSON doc. Esto significa que se está agregando un nuevo par clave-valor al documento JSON.

String jsonOutput;

serializeJson(doc, jsonOutput);

String jsonOutput; Crea una variable de tipo String llamada jsonOutput que servirá para guardar el texto del JSON que se va a generar.

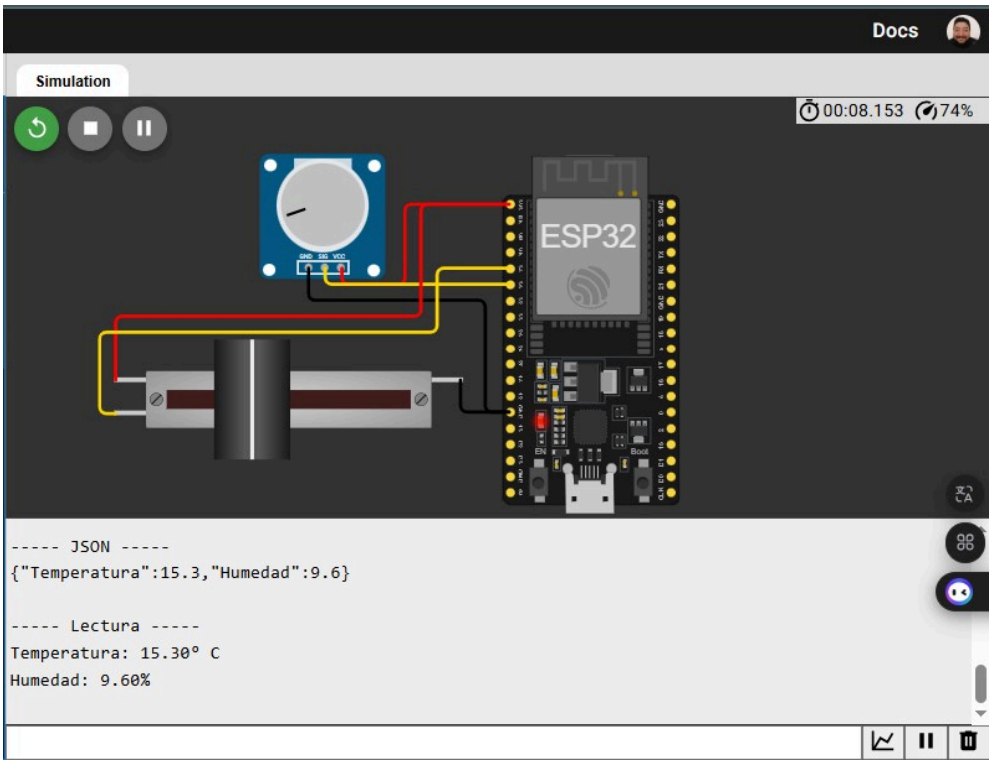
serializeJson(doc, jsonOutput); Convierte el contenido del objeto doc (que es un JSON en memoria) en una cadena de texto. El resultado de esa conversión se guarda dentro de la variable jsonOutput.

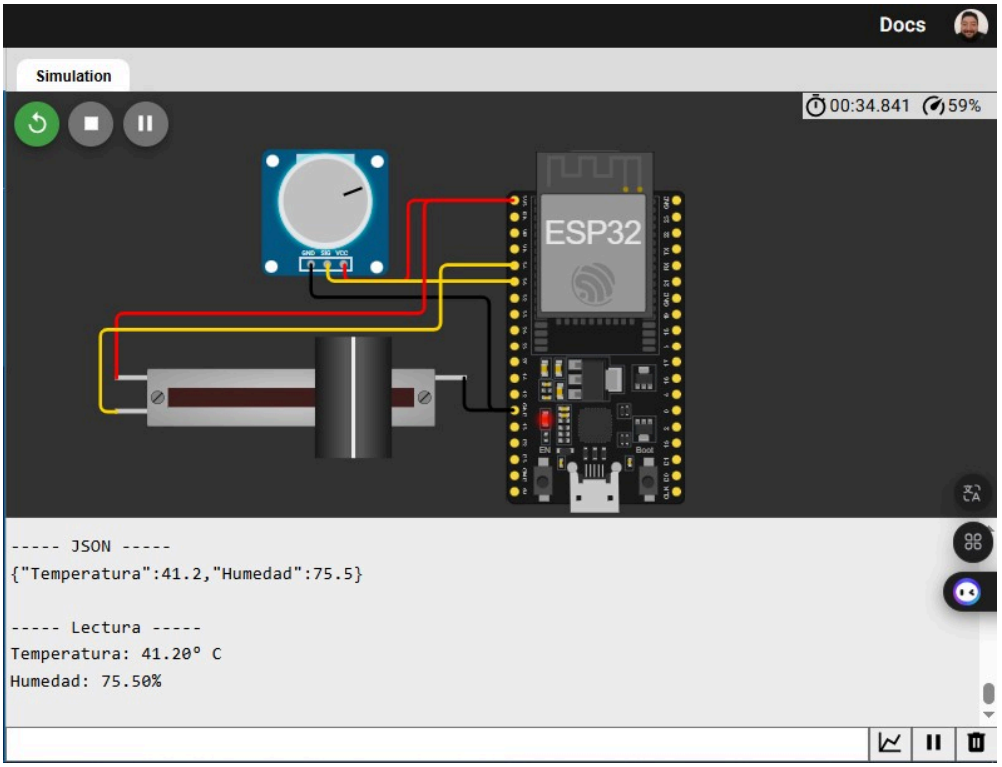
Se comparte el codigo generado en C++ para la simulación en WOKWI, se imprimen en el monitor serial el mensaje JSON y los valores medidos por los potenciometros, simulando los sensores de temperatura y humedad.

AyC TP4-p4

Link a simulación:
https://wokwi.com/projects/428719509160397825

Imagenes de la simulación:





```
#include  <Arduino.h>

#include  <ArduinoJson.h>


const  int tempPin =  34;  // Potenciómetro para temperatura

const  int humPin =  35;  // Potenciómetro para humedad


void  setup()  {

Serial.begin(115200);

}


void  loop()  {


int tempRaw =  analogRead(tempPin);

int humRaw =  analogRead(humPin);


// Convertimos a valores reales

float temperatura =  map(tempRaw,  0,  4095,  0,  500)  /  10.0;  // 0-50 °C

float humedad =  map(humRaw,  0,  4095,  0,  1000)  /  10.0;  // 0-100 %


// Crear objeto JSON

StaticJsonDocument<128> doc;

doc["Temperatura"]  = temperatura;

doc["Humedad"]  = humedad;
```

```
// Serializar como string JSON
String jsonOutput;
serializeJson(doc, jsonOutput);
Serial.println("----- JSON -----");
Serial.println(jsonOutput);

Serial.println();  // Línea en blanco para separación

// Imprimir valores por separado
Serial.println("----- Lectura -----");
Serial.print("Temperatura: ");
Serial.print(temperatura);
Serial.println("° C");

Serial.print("Humedad: ");

Serial.print(humedad);
Serial.println("%");

Serial.println();  // Línea en blanco para separación

delay(3000);
}
```