SERVIDOR WEB PARA CONTROL DE MÚLTIPLES SALIDAS CON FORMATO JSON (INCLUYE INTERRUPTORES DESLIZABLES ESTILIZADOS)

En el primer escenario, desliza el botón para encender GPIO 4. Cuando eso sucede, el navegador realiza una solicitud

HTTP GET en la URL / actualiza? salida = 4 & estado = 1.

xhr_IntercambioParcial.open("GET", "/actualizar?salida="+element.id+"&estado=0", true);

Basado en esa URL, el ESP8266 de Wemos D1 cambia el estado de GPIO 4 a HIGH (1) y enciende el LED. El estado de GPIO se actualiza en el navegador web mediante JavaScript.

En el segundo escenario, desliza el botón para apagar GPIO 4. Cuando eso sucede, el navegador realiza una solicitud

HTTP GET en la URL / actualiza? salida = 4 & estado = 0.

xhr_IntercambioParcial.open("GET", "/actualizar?salida="+element.id+"&estado=0", true);

Según esa URL, cambiamos el estado de GPIO 2 a LOW (0) y apagamos el LED.

LISTA DE MATERIALES:

1 Wemos D1 R1
1 cable USB - micro
1 Protoboard
3 LEDs
3 resistores de 220 Ohms
1 alambre dupont color NEGRO
3 alambres dupont cualquier color diferente a NEGRO.

ARCHIVO main.cpp

```
#include <Arduino.h>
#include <ESP8266WiFi.h>
#include <ESPAsyncTCP.h>
#include <ESPAsyncWebServer.h>
#include <LittleFS.h>
#include <Arduino_JSON.h>

const char* ssid = "MEGACABLE-F7342A";
const char* password = "s345YDeGP";

// Crea el objeto AsyncWebServere en el puerto 80
AsyncWebServer server(80);

//Parámetros de entrada_salida y estado
const char* PARAM_ENTRADA_SALIDA = "salida";
const char* PARAM_ENTRADA_ESTADO = "estado";

// Establece el número de salidas
#define NUM_SALIDAS  4

// Vector con las salidas que se requiere controlar
int salidaGPIOs[NUM_SALIDAS] = {2, 4, 12, 13};
// el pin 13 está en D11/MOSI/D7
// en WEMOS D1 R12 y 14 son LEDs de la tarjeta

// Inicializar LittleFS
void initFS() {
  if (!LittleFS.begin()) {
    Serial.println("Un error ha ocurrido mientras se montaba LittleFS");
  }
  Serial.println("LittleFS montado exitosamente");
}

// Inicializar WiFi
void initWiFi() {
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  Serial.print("Connecting to WiFi ..");
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print('.');
    delay(1000);
  }
  Serial.println(WiFi.localIP());
}

// Devuelve JSON con los estados de salida actuales
String obtenerEstadosDeSalidas() {
```

```
  /* Esta función verifica el estado de todos los GPIOs y devuelve
  una variable string JSON con esa información */
  JSONVar VectorDePuertos; // se crea la variable JSON VectorDePuertos
  // con la información recibida
  for (int i =0; i<NUM_SALIDAS; i++) {
    VectorDePuertos["gpios"][i]["salida"] = String(salidaGPIOs[i]);
    VectorDePuertos["gpios"][i]["estado"] = String(digitalRead(salidaGPIOs[i]));

    /* La variable json VectorDePuertos contiene un arreglo de GPIOs
    con la siguiente estructura:
       gpios[0] = salida = 2 estado = 0
       gpios[1] = salida = 4 estado = 0
       gpios[2] = salida = 12 estado = 0
       gpios[3] = salida = 13 estado = 0
    */
  }
  String cadenaJson = JSON.stringify(VectorDePuertos);
  /* la función JSON.stringfy convierte el vector en formato
  texto JSON */

 Serial.print(cadenaJson); // se envía la cadenaJSON al monitor de arduino
  return cadenaJson;
}

void setup() {
  // Velocidad del puerto serie
  Serial.begin(115200);

  // Prepara GPIOs como salidas
  for (int i =0; i<NUM_SALIDAS; i++) {
    pinMode(salidaGPIOs[i], OUTPUT);
  }

  initWiFi();
  initFS();

  // Ruta de la raíz "/" para la página web
  server.on("/", HTTP_GET, [](AsyncWebServerRequest *request) {
    request->send(LittleFS, "/index.html", "text/html",false);
/* cuando se recibe la petición en el servidor, se envía el texto HTML
grabado en el archivo index.html para construir la página web
En esta ocasió no utilizamos la función procesador porque los estados
son enviados como una respuesta a la petición /estados. Se usa LittleFS
porque se maneja el ESP8266 de Wemos D1. Las placas con Esp 32 manejan
 SPIFFS*/
  });

  server.serveStatic("/", LittleFS, "/");
/* Cuando el archivo HTML se carga en el navegador, realizará una petición
```

```
a los archivos CSS, JavaScript y favicon. Esos son archivos estáticos
grabados en el mismo directorio LittleFS */
  server.on("/quierosaberestados", HTTP_GET, [](AsyncWebServerRequest *request) {
    /* cuando por primera vez se accede a la página web hace una petición sobre
    /quierosaberestados para obtener los estados actuales de los GPIOs
    Esta es la manera en como se envía la cadena JSON como una respuesta */
    String json = obtenerEstadosDeSalidas();
    request->send(200, "application/json", json);
    /* 200 es OK, application/jsomn indica que estamos enviando JSON,
    y 'json' se refiere al contenido de variable que queremos enviar */
    // a continuación grabamos en la variable 'json'
    json = String();
  });

  //Obtener con método GET el valor de salida al
//<ESP_IP>/actualizar?salida=<salida>&estado=<estado>
  server.on("/actualizar", HTTP_GET, [] (AsyncWebServerRequest *request) {
    // se recibe una petición al mover cada switch. 'actualizar' está
    // también en los archivos de JavaScript y JSON
    String salida;
    String estado;
    // Obtener el valor de entrada 1 en <ESP_IP>/actualizar?salida=<salida>&estado=<estado>
    if (request->hasParam(PARAM_ENTRADA_SALIDA) && request-
>hasParam(PARAM_ENTRADA_ESTADO)) {
      // verifica si contiene los parámetros
      salida = request->getParam(PARAM_ENTRADA_SALIDA)->value();
      estado = request->getParam(PARAM_ENTRADA_ESTADO)->value();
      // Controla los correspondientes GPIOs con los correspondientes estados
      digitalWrite(salida.toInt(), estado.toInt());

    }
    else {
      salida = "No hay mensaje enviado";
      estado = "No hay mensaje enviado";
    }
    Serial.print("GPIO: ");
    Serial.print(salida);
    Serial.print(" - Establecido a: ");
    Serial.println(estado);

    request->send(200, "text/plain", "OK");
  });

  // Inicia servidor
  server.begin();
}

void loop() {
}
```

ARCHIVO index.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>Deslizables</title>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" type="text/css" href="style.css">
    <link rel="icon" type="image/png" href="favicon.png">
  </head>
  <body>
    <div class="marquesina">
      <h1>SERVIDOR WEB EN WEMOS D1 - CONTROL DE VARIAS SALIDAS</h1>
    </div>
    <div class="contenido">
      <div class="cuadricula-tarjeta">
        <div class="tarjeta">
          <p class="titulo-tarjeta"> GPIO 2</p>
          <label class="switch">
            <input type="checkbox" onchange="toggleCheckbox(this)" id="2">
            <span class="deslizador"></span>
            <!-- span class Asigna nombres de clases al elemento.
            Por defecto, clases CSS.-->
          </label>
          <p class="estado">Estado: <span id="2s"></span></p>
        </div>
        <div class="tarjeta">
          <p class="titulo-tarjeta"> GPIO 4</p>
          <label class="switch">
            <input type="checkbox" onchange="toggleCheckbox(this)" id="4">
            <span class="deslizador"></span>
            <!-- span class Asigna nombres de clases al elemento.
            Por defecto, clases CSS.-->
          </label>
          <p class="estado">Estado: <span id="4s"></span></p>
        </div>
        <div class="tarjeta">
          <p class="titulo-tarjeta"> GPIO 12</p>
          <label class="switch">
            <input type="checkbox" onchange="toggleCheckbox(this)" id="12">
            <span class="deslizador"></span>
            <!-- span class Asigna nombres de clases al elemento.
            Por defecto, clases CSS.-->
          </label>
          <p class="estado">Estado: <span id="12s"></span></p>
        </div>
        <div class="tarjeta">
          <p class="titulo-tarjeta"> GPIO 13</p>
          <label class="switch">
```

```
          <input type="checkbox" onchange="toggleCheckbox(this)" id="13">
          <span class="deslizador"></span>
          <!-- span class Asigna nombres de clases al elemento.
            Por defecto, clases CSS.-->
        </label>
        <p class="estado">Estado: <span id="13s"></span></p>
      </div>
    </div>
  </div>
  <script src="script.js"></script>
<div align="center"><img src='wemosD1.png'></div>
 </body>
</html>
```

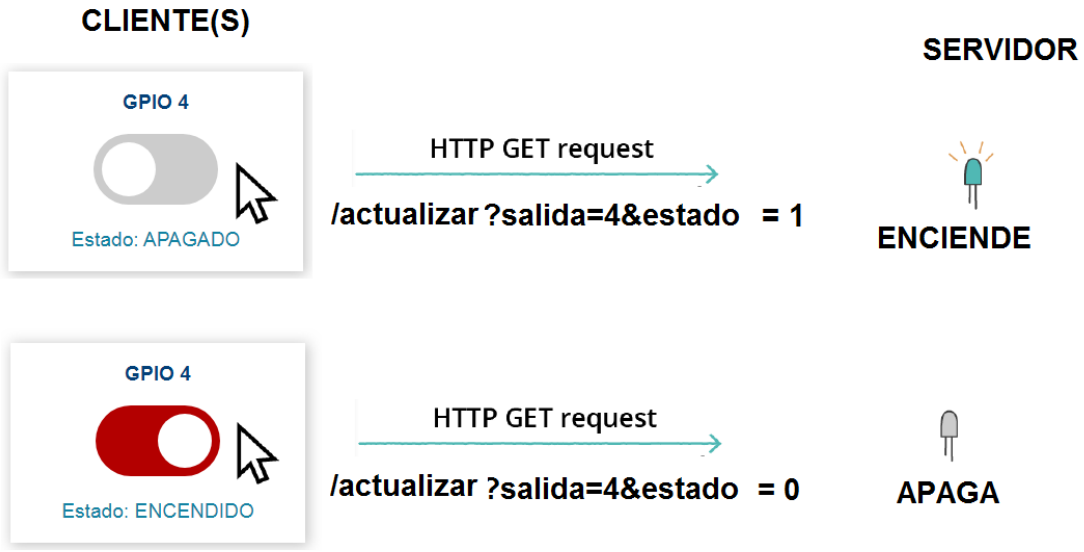
Es necesario copiar el archivo de imagen wemosD1.png al directorio data para que esté junto con los archivos index.html, style.css, script.js y favicon.png

ARCHIVO style.css

```
html { /* observa que los elementos html no tienen punto antes */
  font-family: Arial, Helvetica, sans-serif;
  text-align: center;
}
h1 { /* observa que los elementos html no tienen punto antes */
  font-size: 1.8rem;
  color: white;
}
.marquesina { /* es la barra de arriba */
  overflow: hidden;
  background-color: #0A1128;
}
body { /* observa que los elementos html no tienen punto antes */
  margin: 0;
}
.contenido { /* es el contenido de cada tarjeta */
  padding: 50px;
}
.cuadricula-tarjeta { /* dimensiones y propiedades de la tarjeta */
  max-width: 600px;
  margin: 0 auto;
  display: grid;
  gap: 2rem;
  grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));
}
.tarjeta { /*propiedades de cada tarjeta */
  background-color: white;
  box-shadow: 2px 2px 12px 1px rgba(140,140,140,.5);
}
.titulo-tarjeta { /* título de cada tarjeta */
  font-size: 1.2rem;
```

```
  font-weight: bold;
  color: #034078
}
.estado { /* propiedades del mensaje de estado de los GPIO */
  font-size: 1.2rem;
  color:#1282A2;
}
.switch { /* propiedades de cada switch */
  position: relative;
  display: inline-block;
  width: 120px;
  height: 68px
}
.switch input { /* cuando "no se ve" el switch */
  display: none
}
.deslizador { /* características cuando "no se ve" el switch */
  position: absolute;
  top: 0; left: 0; right: 0; bottom: 0;
  background-color: #ccc;
  border-radius: 50px
}
.deslizador:before {
  position: absolute;
  content: "";
  height: 52px;
  width: 52px;
  left: 8px;
  bottom: 8px;
  background-color: #fff;
  transition: .4s;
  border-radius: 50px;
}
input:checked+.deslizador {  /* color al deslizar ENCENDIDO */
  background-color: #b30000;
}
input:checked+.deslizador:before {
  -webkit-transform: translateX(52px);
  -ms-transform: translateX(52px);
  transform: translateX(52px);
}
```

ARCHIVO script.js



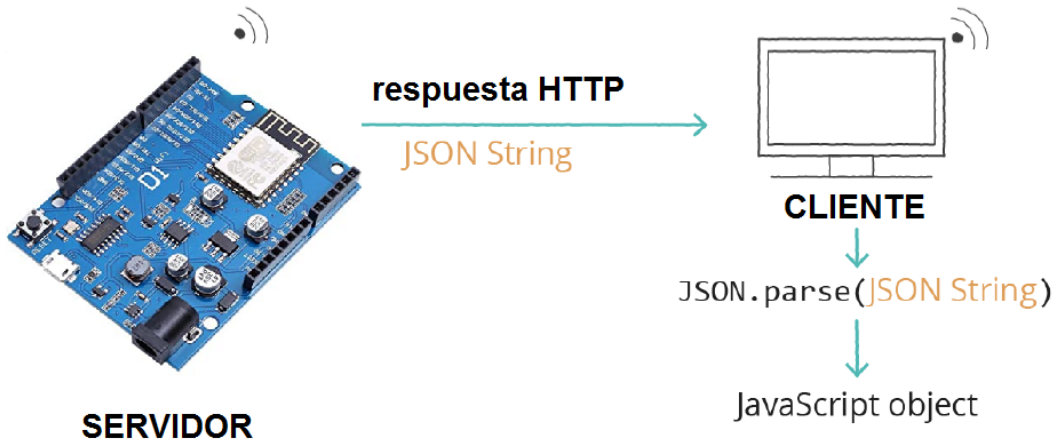
FUNCIÓN DE CONVERSIÓN JSON.parse

JSON es un estándar para almacenar e intercambiar datos en una forma conveniente a servidores y a clientes. Significa **J**ava**S**cript **O**bjet **N**otation.

JSON es texto y cualquier objeto JavaScript puede ser convertido en texto JSON y enviarse al servidor. El servidor puede enviar texto **JSON** al cliente que puede ser convertido en objeto de JavaScript.

La sintaxis JSON es:

- Los datos son representados en pares nombre/valor.
- Cada nombre es seguido por dos puntos (:);
- Los nombres/valores son separados por comas.
- Las llaves abarcan objetos.
- Los corchetes abarcan arreglos.



```
// Se obtienen los estados actuales de los GPIOs cuando la pagina carga
window.addEventListener('load', obtieneEstados);

// Función para obtener y actualizar los estados de los GPIOs
// en la página web cuando carga por primera ocasión.

function obtieneEstados(){
  var xhr_intercambiadatossinrecargartodalapagina = new XMLHttpRequest();

  // se crea el objeto para intercambiar datos parciales de la página web
  xhr_intercambiadatossinrecargartodalapagina.onreadystatechange = function() {

if (this.readyState == 4 && this.status == 200)
 {
  var miObjetoProvenienteDeTextoJSON = JSON.parse(this.responseText);
      /* JSON.parse() es una función interna de JavaScript para convertir una
cadena escrita en formato JSON a objetos nativos de JavaScript */
      console.log(miObjetoProvenienteDeTextoJSON);

 for (i in miObjetoProvenienteDeTextoJSON.gpios) {
        var salida = miObjetoProvenienteDeTextoJSON.gpios[i].salida;
        var estado = miObjetoProvenienteDeTextoJSON.gpios[i].estado;
        console.log(salida);
        console.log(estado);

 // podemos observar salida y estado de cada GPIO en consola
        if (estado == "1") {
          document.getElementById(salida).checked = true;
          document.getElementById(salida+"s").innerHTML = "ENCENDIDO";
         /*  con la "s" concatenada a salida,
        necesitamos actualizar el texto de estado a "ENCENDIDO" */
        }
        else {
          document.getElementById(salida).checked = false;
          document.getElementById(salida+"s").innerHTML = "APAGADO";
           /*  con la "s" concatenada a salida,
        necesitamos actualizar el texto de estado a "APAGADO" */
        }
      }
    }
  };
  xhr_intercambiadatossinrecargartodalapagina.open("GET", "/quierosaberestados", true);
  xhr_intercambiadatossinrecargartodalapagina.send();
  // Envía las salidas y los estados actuales del los GPIOs
}

// Envía peticiones para controlar GPIOs
function toggleCheckbox (element) {
```

```
  /* Función de JavaScript que envía peticiones al ESP8266 de la Wemos D1
     cuando hacemos clic en los switches deslizadores, esta función
     acepta como argumento el id de le checkboox */

  var xhr_intercambiadatossinrecargartodalapagina = new XMLHttpRequest();

  if (element.checked) {

 xhr_intercambiadatossinrecargartodalapagina.open("GET",
"/actualizar?salida="+element.id+"&estado=1", true);
     /* se recibe una petición al mover cada switch. 'actualizar' está
      también en los archivos de main.cpp y script.json */
     document.getElementById(element.id+"s").innerHTML = "ENCENDIDO";
     /*  con la "s" concatenada a salida,
         necesitamos actualizar el texto de estado a "ENCENDIDO" */
  }
  else {

 xhr_intercambiadatossinrecargartodalapagina.open("GET",
"/actualizar?salida="+element.id+"&estado=0", true);
     document.getElementById(element.id+"s").innerHTML = "APAGADO";
     /*  con la "s" concatenada a salida,
         necesitamos actualizar el texto de estado a "APAGADO" */
  }

 xhr_intercambiadatossinrecargartodalapagina.send();
  // Envía las salidas y los estados actuales del los GPIOs
}
```

ARCHIVO PLATFORMIO.INI

Una vez que se han añadido las libraries al proyecto, el archivo platformio.ini debe quedar de la siguiente manera:

```
[env:d1]
platform = espressif8266
board = d1
framework = arduino
monitor_speed = 115200
lib_deps =
        ESP Async ESPAsyncWebServer
        ardduino-libraries/Arduino_JSON @ 0.1.0
        arduino-libraries/Arduino_JSON@^0.1.0
board_build.filesystem = littlefs
```

CONEXIONES

Conectaremos LEDs a los siguientes pines de la tarjeta Wemos D1. Cada LED debe incluir un resistor de 220 Ohms conectado en serie y hacia tierra:

D2 (Led incluido en la tarjeta Wemos D1)
D4 en el pin etiquetado como /D4
D12 en el pin etiquetado como /D12
D13 Conectar LED en el pin etiquetado como /D7 de la Wemos D1.

PRUEBA:

Al telear la IP proporcionada por el monitor de Platformio, debe aparecer en el navegador lo siguiente

OBSERVACIÓN EN CONSOLA

Al estarse ejecutando el server en la Wemos, en el cliente mediante el navegador podemos teclear Ctrol-Shift-J y se observará lo siguiente en consola: