```
#include <ESP8266WiFi.h>

// Definicion de los gpios de los que dispone el ESP-01
// GPIO 2 para la fuente
// GPIO 0 para la luz
const int font = 2;
const int light = 0;

// Variables auxiliares para guardar el estado actual de la salida
String fontState = "off";
String lightState = "off";


// Configuracion de IP estatica
IPAddress ip(192, 168, 0, 10);
IPAddress subnet(255, 255, 255, 0);
IPAddress gateway(192, 168, 0, 1);
IPAddress dns(9, 9, 9, 9);

// SSID y Pass de la conexion WLAN
const char* ssid     = "Tu_WIFI";
const char* password = "Tu_Password";

WiFiServer server(80);

// Variable para almacencar las "HTTP request"
String header;

void wifi_conf()
{
 WiFi.mode(WIFI_STA);

 WiFi.config(ip, gateway, subnet, dns);

 delay(100);

 WiFi.begin(ssid, password);
 Serial.print("Conectando a la wifi ...");

 while (WiFi.status() != WL_CONNECTED)
 {
   delay(500);
   Serial.print(".");
 }

 while (WiFi.waitForConnectResult() != WL_CONNECTED)
 {
   Serial.println();
   Serial.println("Fallo en la conexion wlan");
   delay(5000);
   ESP.restart();
 }

 Serial.println();
 Serial.print("Conectado con la direccion IP: ");
 Serial.println(WiFi.localIP());
}


void setup()
{
 Serial.begin(115200);
 Serial.println();

 pinMode(font, OUTPUT);
 digitalWrite(font, LOW);

 pinMode(light, OUTPUT);
 digitalWrite(light, LOW);

 wifi_conf();

 server.begin();
}


void loop()
{
 WiFiClient client = server.available();  // Listen for incoming clients

 if (client) {                             // If a new client connects,
   Serial.println("New Client.");          // print a message out in the serial port
   String currentLine = "";                // make a String to hold incoming data from the client
   while (client.connected()) {            // loop while the client's connected
     if (client.available()) {             // if there's bytes to read from the client,
       char c = client.read();             // read a byte, then
       Serial.write(c);                    // print it out the serial monitor
       header += c;
       if (c == '\n') {                    // if the byte is a newline character
         // if the current line is blank, you got two newline characters in a row.
         // that's the end of the client HTTP request, so send a response:
         if (currentLine.length() == 0) {
           // HTTP headers always start with a response code (e.g. HTTP/1.1 200 OK)
           // and a content-type so the client knows what's coming, then a blank line:
           client.println("HTTP/1.1 200 OK");
           client.println("Content-type:text/html");
           client.println("Connection: close");
           client.println();

           // turns the GPIOs on and off
           if (header.indexOf("GET /2/on") >= 0) {
             Serial.println("GPIO 2 on");
             fontState = "on";
             digitalWrite(font, HIGH);
           } else if (header.indexOf("GET /2/off") >= 0) {
             Serial.println("GPIO 2 off");
             fontState = "off";
             digitalWrite(font, LOW);
           } else if (header.indexOf("GET /0/on") >= 0) {
             Serial.println("GPIO 0 on");
             lightState = "on";
             digitalWrite(light, HIGH);
           } else if (header.indexOf("GET /0/off") >= 0) {
             Serial.println("GPIO 0 off");
             lightState = "off";
             digitalWrite(light, LOW);
           }

           // Display the HTML web page
           client.println("<!DOCTYPE html><html>");
           client.println("<head><meta name=\"viewport\" content=\"width=device-width, initial-scale=1\">");
           client.println("<link rel=\"icon\" href=\"data:,\">");
           // CSS to style the on/off buttons
           // Feel free to change the background-color and font-size attributes to fit your preferences
           client.println("<style>html { font-family: Helvetica; display: inline-block; margin: 0px auto; text-align: center;}");
           client.println(".button { background-color: #195B6A; border: none; color: white; padding: 16px 40px;");
           client.println("text-decoration: none; font-size: 30px; margin: 2px; cursor: pointer;}");
           client.println(".button2 {background-color: #77878A;}</style></head>");

           // Web Page Heading
           client.println("<body><h1>ESP8266 Web Server</h1>");

           // Display current state, and ON/OFF buttons for GPIO 2
           client.println("<p>GPIO 2 - State " + fontState + "</p>");
           // If the fontState is off, it displays the ON button
           if (fontState=="off") {
            client.println("<p><a href=\"/2/on\"><button class=\"button\">ON</button></a></p>");
            } else {
            client.println("<p><a href=\"/2/off\"><button class=\"button button2\">OFF</button></a></p>");
            }

           // Display current state, and ON/OFF buttons for GPIO 0
           client.println("<p>GPIO 0 - State " + lightState + "</p>");
           // If the lightState is off, it displays the ON button
           if (lightState=="off") {
            client.println("<p><a href=\"/0/on\"><button class=\"button\">ON</button></a></p>");
            } else {
            client.println("<p><a href=\"/0/off\"><button class=\"button button2\">OFF</button></a></p>");
            }
           client.println("</body></html>");

           // The HTTP response ends with another blank line
           client.println();
           // Break out of the while loop
           break;
         } else { // if you got a newline, then clear currentLine
           currentLine = "";
         }
       } else if (c != '\r') {  // if you got anything else but a carriage return character,
         currentLine += c;      // add it to the end of the currentLine
       }
     }
   }
   // Clear the header variable
   header = "";
   // Close the connection
   client.stop();
   Serial.println("Client disconnected.");
   Serial.println("");
 }
}
```