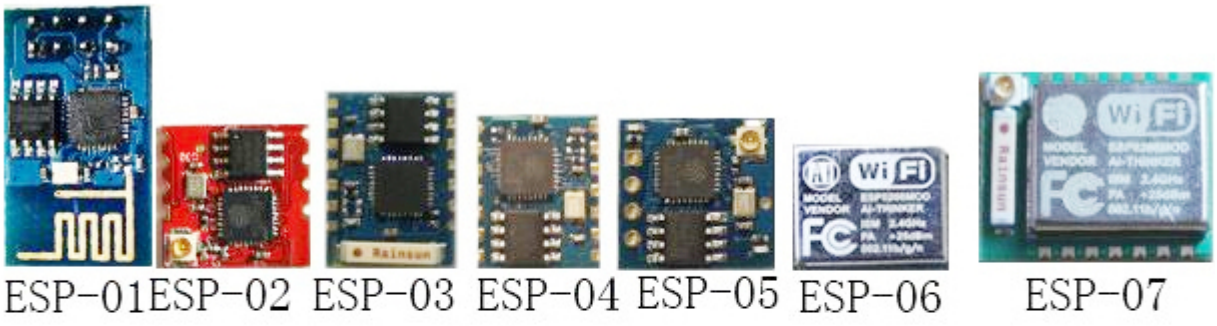
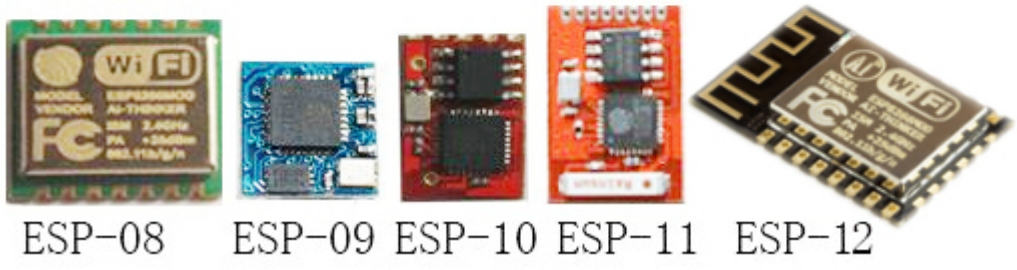
ESP8266 ESP-01 e adattatore USB CH340

(Prof Fischetti P.)

Prestare attenzione alla tensione delle linee del segnale, poiché ESP8266 funziona solo con logica a 3.3v e i pin non tollerano i 5 V.

L' ESP8266 è prodotto dalla Cinese Espressif, contiene un microcontrollore, un modulo WiFi e dei pin GPIO personalizzabili. Esistono in commercio numerose varianti che si differenziano tra loro per le caratteristiche costruttive: il tipo di antenna del WiFi, la certificazione FCC o meno, la quantità di memoria, la presenza o meno di adattatori usb/seriali a bordo, il numero di Pin attivi.





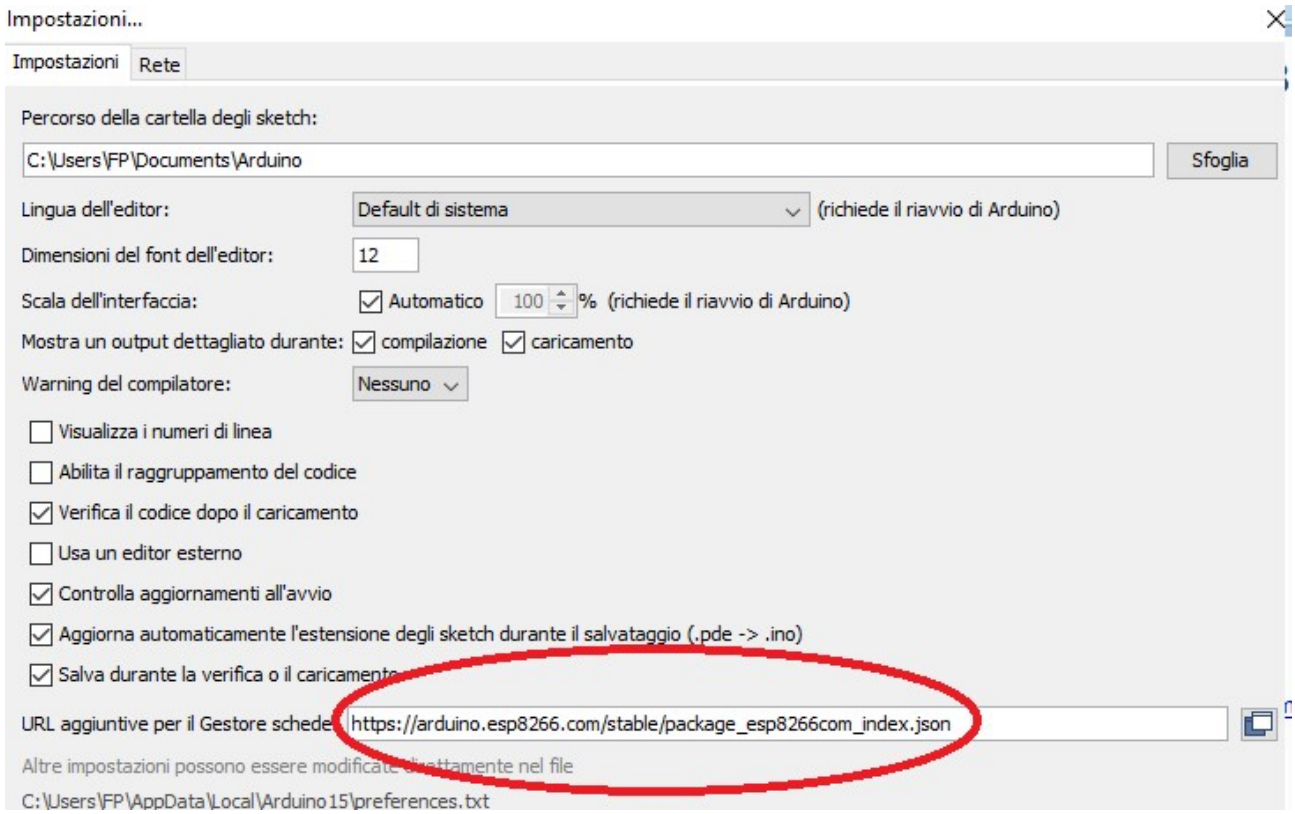
ESP-01

Questa è la prima e più semplice scheda che utilizza ESP8266. Permette di allegare linee seriali e scomporre solo due pin GPIO per uso nativo. Questo è anche il più economico e può essere acquistato da molti fornitori cinesi a 2,5 dollari.

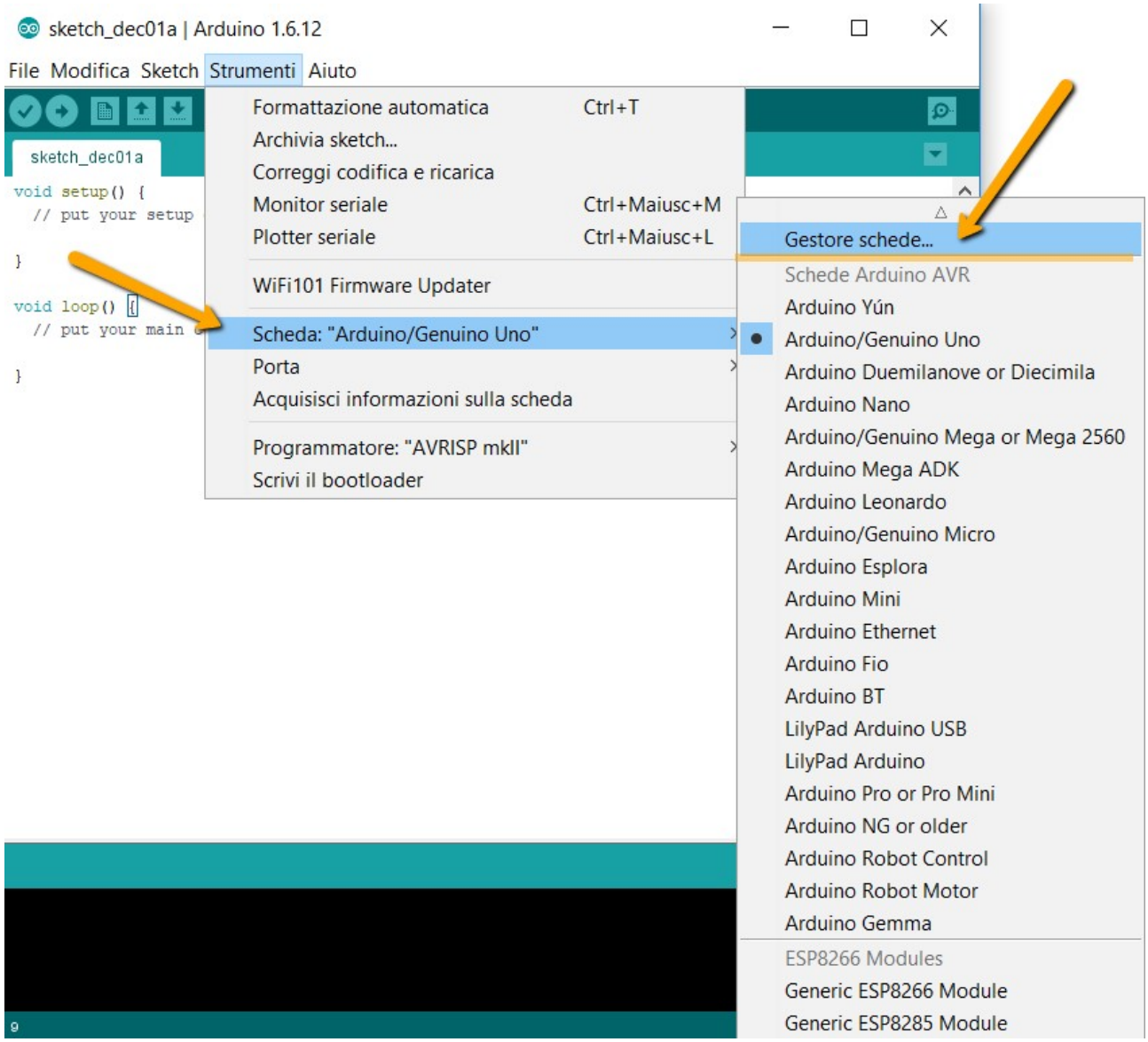
Driver per Windows10: CH341SER.EXE (http://www.wch-ic.com/downloads/CH341SER_EXE.html)

IDE Arduino: Installazione Scheda ESP8266

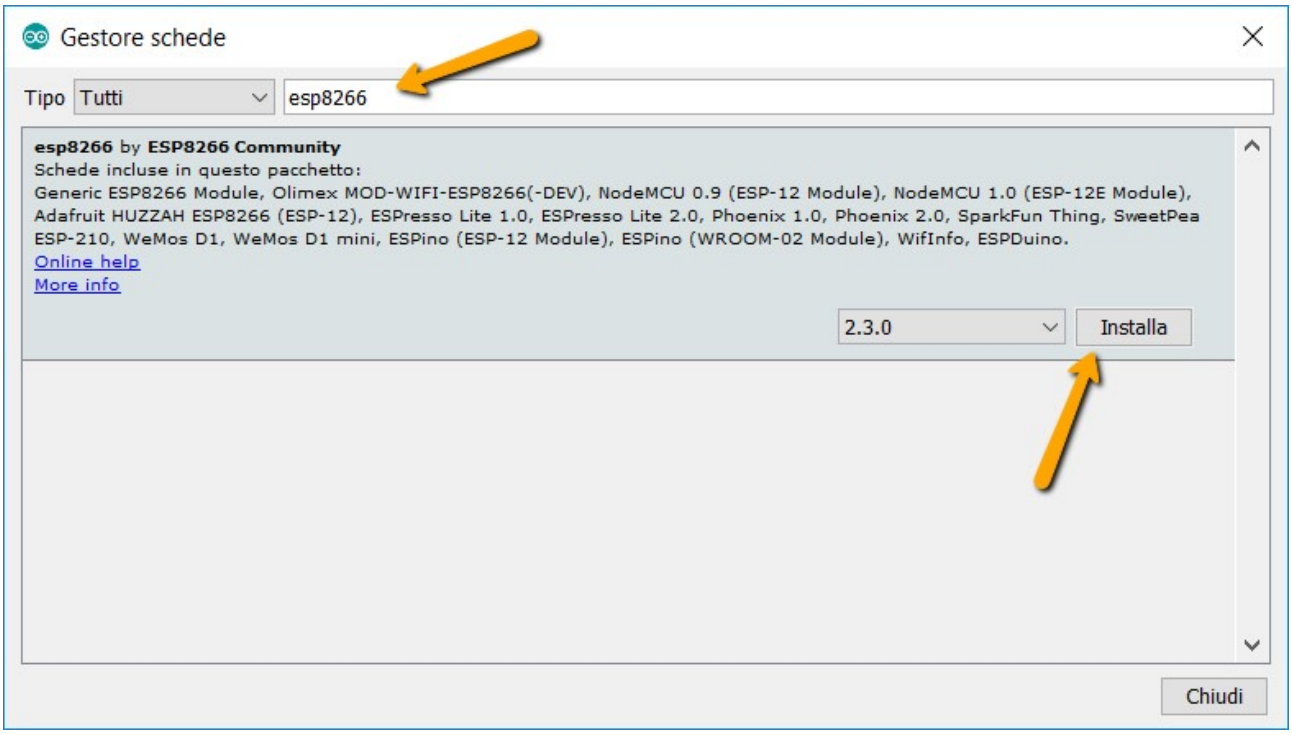
IDE>File>Impostazioni:Aggiungere l'URL
http://arduino.esp8266.com/stable/package_esp8266com_index.json



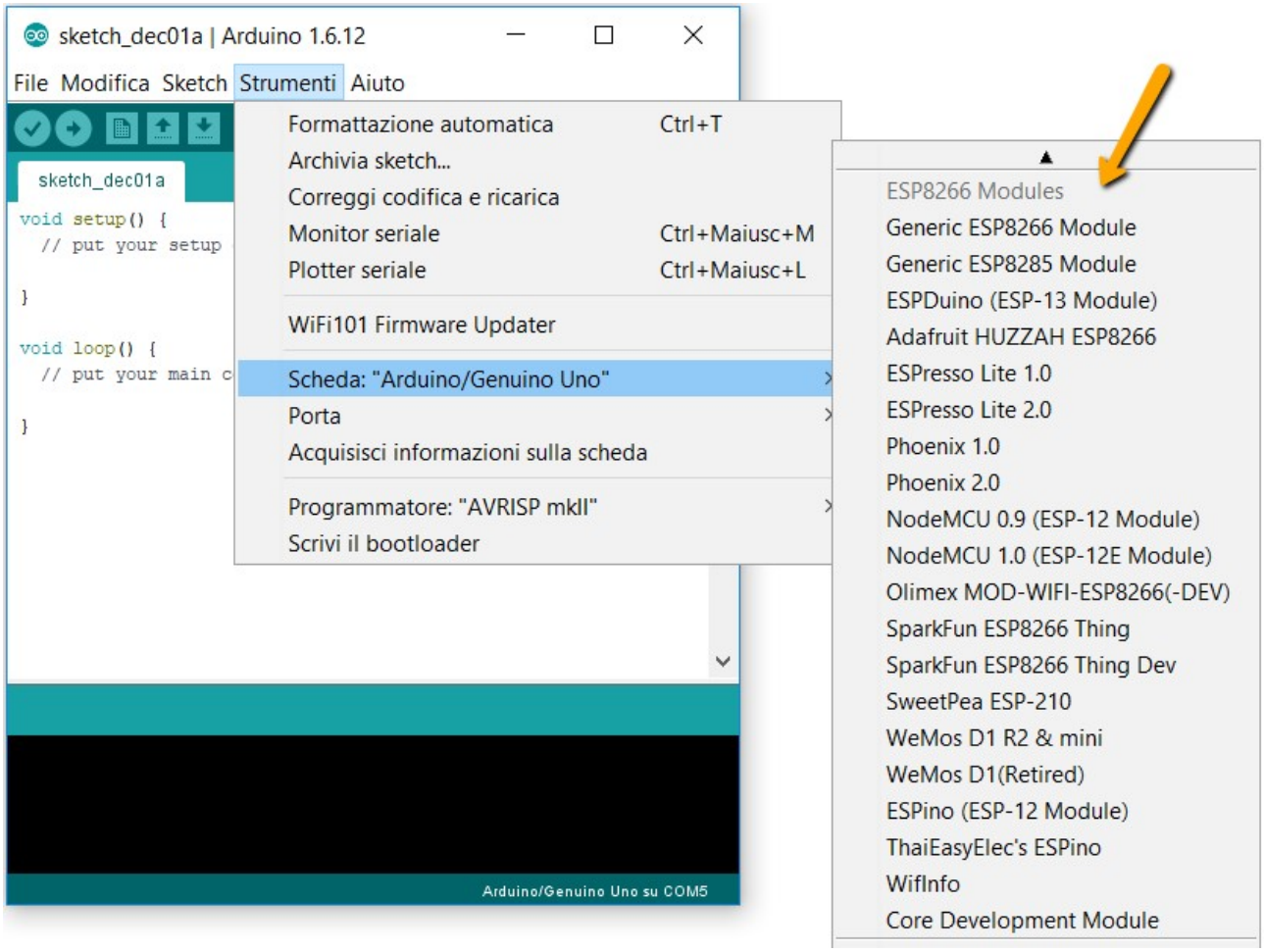
IDE>Strumenti>Gestore Schede



Cerca ESP8266>Installa:

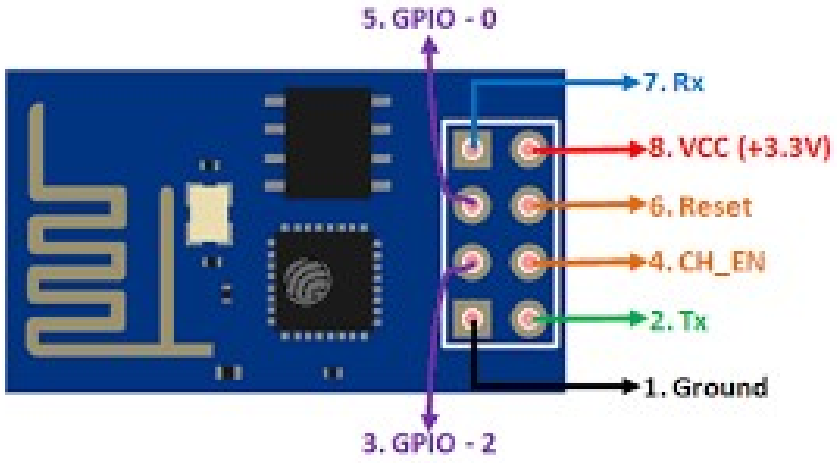


Se tutto OK Selezinare:Strumenti>Scheda "…">Generic ESP8266 Module



Configurazione scheda ESP8266

Scheda: "Generic ESP8266 Module"
Flash Mode: "DOUT (compatible)"
Flash Size: "1MB (FS:64KB OTA:~470KB)"
Debug port: "Disabled"
Debug Level: "Nessuno"
lwIP Variant: "v2 Lower Memory"
Non-32-Bit Access: "Use pgm_read macros for IRAM/PROGMEM"
Reset Method: "no dtr (aka ck)"
Crystal Frequency: "26 MHz"
SSL Support: "All SSL ciphers (most compatible)"
VTables: "Flash"
Espressif FW: "nonos-sdk 2.2.1+100 (190703)"
Flash Frequency: "40MHz"
MMU: "32KB cache + 32KB IRAM (balanced)"
CPU Frequency: "80 MHz"
C++ Exceptions: "Disabled (new aborts on oom)"
Stack Protection: "Disabled"
Builtin Led: "2"
Upload Speed: "115200"
Erase Flash: "Only Sketch"
Porta: "COM15"



Collegare il PIN 1 (Ground) con il PIN 5 (GPIO-0) con un pulsante o Filo

Inserire il dispositivo montato nell'USB del PC

Dopo pochi istanti staccare il collegamento precedente tra Pin 1 e 5

Scrivere e caricare lo sketch:

/*********

```
  Rui Santos
  Complete project details at https://randomnerdtutorials.com
*********/

// Load Wi-Fi library
#include <ESP8266WiFi.h>

// Replace with your network credentials
const char* ssid     = "<AP Name>";
const char* password = "< AP pwd>";

// Set web server port number to 80
WiFiServer server(80);

// Variable to store the HTTP request
String header;

// Auxiliar variables to store the current output state
String output5State = "off";
String output4State = "off";

// Assign output variables to GPIO pins
const int output5 = 5;
const int output4 = 4;

// Current time
unsigned long currentTime = millis();
// Previous time
unsigned long previousTime = 0;
// Define timeout time in milliseconds (example: 2000ms = 2s)
const long timeoutTime = 2000;

void setup() {
  Serial.begin(9600);
  // Initialize the output variables as outputs
  pinMode(output5, OUTPUT);
  pinMode(output4, OUTPUT);
  // Set outputs to LOW
  digitalWrite(output5, LOW);
  digitalWrite(output4, LOW);

  // Connect to Wi-Fi network with SSID and password
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  // Print local IP address and start web server
  Serial.println("");
  Serial.println("WiFi connected.");
  Serial.println("IP address: ");
```

```
  Serial.println(WiFi.localIP());
  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
    currentTime = millis();
    previousTime = currentTime;
    while (client.connected() && currentTime - previousTime <= timeoutTime) { // loop while the client's
connected
      currentTime = millis();
      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 /5/on") >= 0) {
              Serial.println("GPIO 5 on");
              output5State = "on";
              digitalWrite(output5, HIGH);
            } else if (header.indexOf("GET /5/off") >= 0) {
              Serial.println("GPIO 5 off");
              output5State = "off";
              digitalWrite(output5, LOW);
            } else if (header.indexOf("GET /4/on") >= 0) {
              Serial.println("GPIO 4 on");
              output4State = "on";
              digitalWrite(output4, HIGH);
            } else if (header.indexOf("GET /4/off") >= 0) {
              Serial.println("GPIO 4 off");
              output4State = "off";
              digitalWrite(output4, LOW);
            }

            // Display the HTML web page
            client.println("<!DOCTYPE html><html>");
            client.println("<head><meta name=\"viewport\" content=\"width=device-width, initial-
scale=1\">");
```

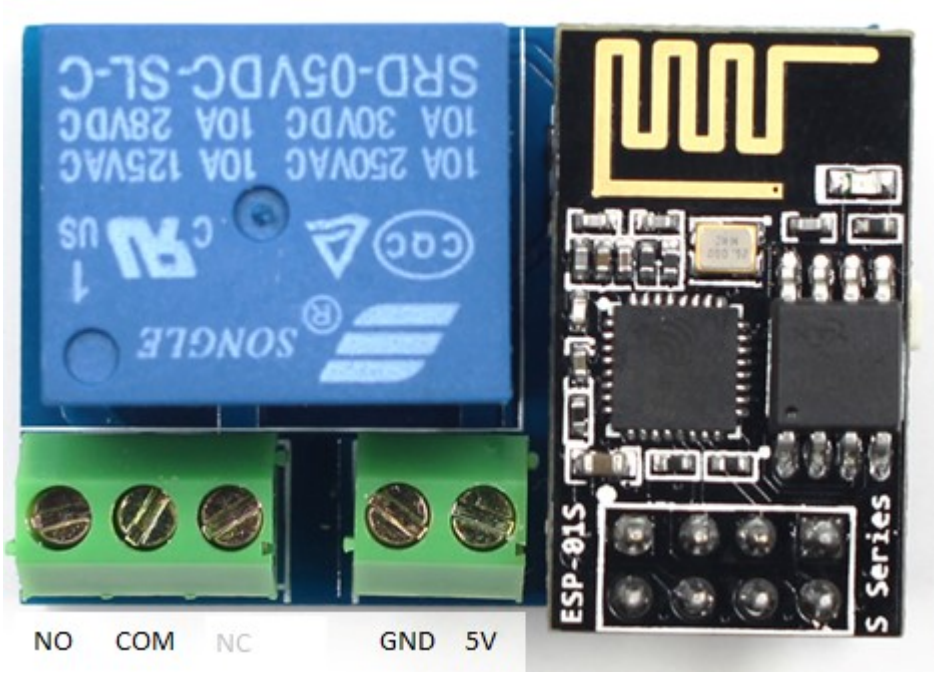
```
            client.println("<link rel=\"icon\" href=\"data:,\">");
            // CSS to style the on/off buttons
            // Feel free to change the background-color and font-size attributes to fit your preferences
            client.println("<style>html { font-family: Helvetica; display: inline-block; 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 5
            client.println("<p>GPIO 5 - State " + output5State + "</p>");
            // If the output5State is off, it displays the ON button
            if (output5State=="off") {
              client.println("<p><a href=\"/5/on\"><button class=\"button\">ON</button></a></p>");
            } else {
              client.println("<p><a href=\"/5/off\"><button class=\"button button2\">OFF</button></a></p>");
            }

            // Display current state, and ON/OFF buttons for GPIO 4
            client.println("<p>GPIO 4 - State " + output4State + "</p>");
            // If the output4State is off, it displays the ON button
            if (output4State=="off") {
              client.println("<p><a href=\"/4/on\"><button class=\"button\">ON</button></a></p>");
            } else {
              client.println("<p><a href=\"/4/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("");
  }
}
```




```
#include <ESP8266WiFi.h>

const char* ssid     = "<AP Name>";
const char* password = "< AP pwd>";

 #define RELAY 0 // relay connected to  GPIO0
WiFiServer server(80);

void setup()
{
  Serial.begin(9600); // must be same baudrate with the Serial Monitor

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

  // Connect to WiFi network
  Serial.println();
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

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

```
  // Start the server
  server.begin();
  Serial.println("Server started");

  // Print the IP address
  Serial.print("Use this URL to connect: ");
  Serial.print("https://192.168.0.178/");
  Serial.print(WiFi.localIP());
  Serial.println("/");

}

void loop()
{
  // Check if a client has connected
  WiFiClient client = server.available();
  if (!client)
  {
    return;
  }

  // Wait until the client sends some data
  Serial.println("new client");
  while(!client.available())
  {
    delay(1);
  }

  // Read the first line of the request
  String request = client.readStringUntil('\r');
  Serial.println(request);
  client.flush();

  // Match the request
  int value = LOW;
  if (request.indexOf("/RELAY=ON") != -1)
  {
    Serial.println("RELAY=ON");
    digitalWrite(RELAY,LOW);
    value = LOW;
  }
  if (request.indexOf("/RELAY=OFF") != -1)
  {
    Serial.println("RELAY=OFF");
    digitalWrite(RELAY,HIGH);
    value = HIGH;
  }

  // Return the response
  client.println("HTTP/1.1 200 OK");
  client.println("Content-Type: text/html");
  client.println(""); //  this is a must
```

```
 client.println("<!DOCTYPE HTML>");
 client.println("<html>");
 client.println("<head><title>ESP8266 RELAY Control</title><link rel='shortcut icon' href='#'></head>");
 client.print("Relay is now: ");

 if(value == HIGH)
 {
  client.print("OFF");
 }
 else
 {
  client.print("ON");
 }
 client.println("<br><br>");
 client.println("Turn <a href=\"/RELAY=OFF\">OFF</a> RELAY<br>");
 client.println("Turn <a href=\"/RELAY=ON\">ON</a> RELAY<br>");
  client.println("</html>");

 delay(1);
 Serial.println("Client disconnected");
 Serial.println("");
}
```

N.B. In rosso il codice per rimuovere la richiesta ‘favicon’.