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

// Replace with your network credentials
const char* ssid     = "Enter wifi user id";
const char* password = "Enter password";

// 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";
String output16State = "off";
String output2State = "off";
String output0State = "off";
String output14State = "off";
String output12State = "off";
String output13State = "off";

// Assign output variables to GPIO pins
const int output5 = 5;
const int output4 = 4;
const int output16 = 16;
const int output2 = 2;
const int output0 = 0;
```

```
const int output14 = 14;
const int output12 = 12;
const int output13 = 13;


// 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(115200);
  // Initialize the output variables as outputs
  pinMode(output5, OUTPUT);
  pinMode(output4, OUTPUT);
  pinMode(output16, OUTPUT);
 pinMode(output2, OUTPUT);
  pinMode(output0, OUTPUT);
  pinMode(output14, OUTPUT);
   pinMode(output12, OUTPUT);
  pinMode(output13, OUTPUT);

  // Set outputs to LOW
  digitalWrite(output5, LOW);
  digitalWrite(output4, LOW);
  digitalWrite(output16, LOW);
  digitalWrite(output2, LOW);
  digitalWrite(output0, LOW);
  digitalWrite(output14, LOW);
```

```
  digitalWrite(output12, LOW);
  digitalWrite(output13, 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);
      }



      // turns the GPIOs on and off
      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);
}


else if (header.indexOf("GET /16/on") >= 0) {
  Serial.println("GPIO 16 on");
  output16State = "on";
  digitalWrite(output16, HIGH);
}

else if (header.indexOf("GET /16/off") >= 0) {
  Serial.println("GPIO 16 off");
  output16State = "off";
  digitalWrite(output16, LOW);
}

else if (header.indexOf("GET /0/on") >= 0) {
  Serial.println("GPIO 0 on");
  output0State = "on";
  digitalWrite(output0, HIGH);
}

else if (header.indexOf("GET /0/off") >= 0) {
```

```
  Serial.println("GPIO 0 off");
  output0State = "off";
  digitalWrite(output0, LOW);
}
else if (header.indexOf("GET /2/on") >= 0) {
  Serial.println("GPIO 2 on");
  output2State = "on";
  digitalWrite(output2, HIGH);
}

else if (header.indexOf("GET /2/off") >= 0) {
  Serial.println("GPIO 2 off");
  output2State = "off";
  digitalWrite(output2, LOW);
}



else if (header.indexOf("GET /14/off") >= 0) {
  Serial.println("GPIO 14 off");
  output14State = "off";
  digitalWrite(output14, LOW);
}

else if (header.indexOf("GET /14/on") >= 0) {
  Serial.println("GPIO 14 on");
  output14State = "on";
  digitalWrite(output14, HIGH);
}

else if (header.indexOf("GET /12/off") >= 0) {
  Serial.println("GPIO 12 off");
```

```
            output12State = "off";
            digitalWrite(output12, LOW);
          }

          else if (header.indexOf("GET /12/on") >= 0) {
            Serial.println("GPIO 12 on");
            output12State = "on";
            digitalWrite(output12, HIGH);
          }

          else if (header.indexOf("GET /13/off") >= 0) {
            Serial.println("GPIO 13 off");
            output13State = "off";
            digitalWrite(output13, LOW);
          }

          else if (header.indexOf("GET /13/on") >= 0) {
            Serial.println("GPIO 13 on");
            output13State = "on";
            digitalWrite(output13, HIGH);
          }



          // 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:
12px 40px;");
            client.println("text-decoration: none; font-size: 20px; margin: 0px; cursor: pointer;}");
            client.println(".button2 {background-color: #77878A;}</style></head>");

            // Web Page Heading
            client.println("<body><h3>ESP8266 Web Server</h3>");

            // 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>");
            }

            // Display current state, and ON/OFF buttons for GPIO 16
            client.println("<p>GPIO 16 - State " + output16State + "</p>");
            // If the output4State is off, it displays the ON button
            if (output16State=="off") {
```

```
            client.println("<p><a href=\"/16/on\"><button class=\"button\">ON</button></a></p>");
          } else {
            client.println("<p><a href=\"/16/off\"><button class=\"button button2\">OFF</button></a></p>");
          }


 // Display current state, and ON/OFF buttons for GPIO 2
          client.println("<p>GPIO 2 - State " + output2State + "</p>");
          // If the output4State is off, it displays the ON button
          if (output2State=="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 " + output0State + "</p>");
          // If the output4State is off, it displays the ON button
          if (output0State=="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>");
          }


 // Display current state, and ON/OFF buttons for GPIO 14
```

```
            client.println("<p>GPIO 14 - State " + output14State + "</p>");
            // If the output4State is off, it displays the ON button
            if (output14State=="off") {
              client.println("<p><a href=\"/14/on\"><button class=\"button\">ON</button></a></p>");
            } else {
              client.println("<p><a href=\"/14/off\"><button class=\"button button2\">OFF</button></a></p>");
            }

// Display current state, and ON/OFF buttons for GPIO 12
            client.println("<p>GPIO 12 - State " + output12State + "</p>");
            // If the output4State is off, it displays the ON button
            if (output12State=="off") {
              client.println("<p><a href=\"/12/on\"><button class=\"button\">ON</button></a></p>");
            } else {
              client.println("<p><a href=\"/12/off\"><button class=\"button button2\">OFF</button></a></p>");
            }


// Display current state, and ON/OFF buttons for GPIO 13
            client.println("<p>GPIO 13 - State " + output13State + "</p>");
            // If the output4State is off, it displays the ON button
            if (output13State=="off") {
              client.println("<p><a href=\"/13/on\"><button class=\"button\">ON</button></a></p>");
            } else {
              client.println("<p><a href=\"/13/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("");
 }
}
```

----------End----------

Load to Nodemcu

Run it.

Go to Monitor

Read IP Address

Now enter in browser – same ip address

Website opens with switch ON/OFF commands

Click and see LED is getting ON/OFF .

LED should be connected to Nodemcu PIN as follows –

pinMode(output5, OUTPUT);

pinMode(output4, OUTPUT);

pinMode(output16, OUTPUT);

pinMode(output2, OUTPUT);

pinMode(output0, OUTPUT);

pinMode(output14, OUTPUT);

pinMode(output12, OUTPUT);

pinMode(output13, OUTPUT);

Dinesh Kumar (ISRO, Bangalore)