Name: Yash Beniwal

Id:201951175

Experiment - 03

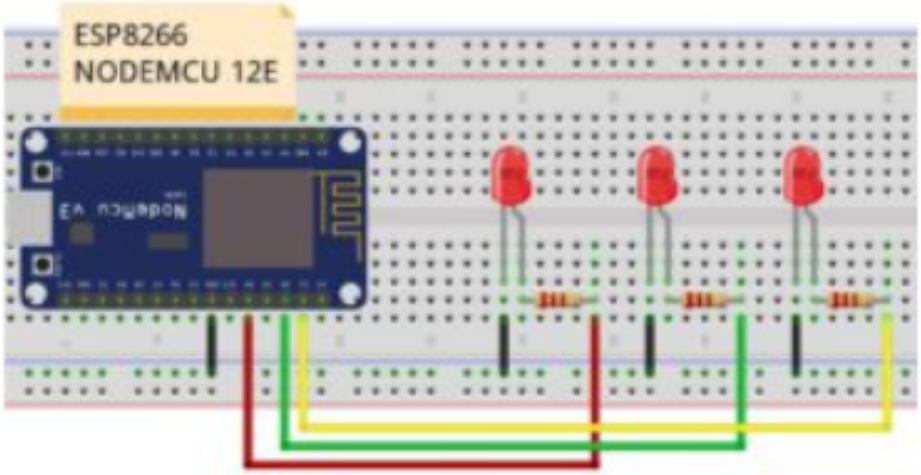
1. Objective: Using ESP8266 make your own LED Control web server in Arduino IDE

Day by day IoT devices are increasing. Because it gives you comfort, you can literally control any device from anywhere. You just need an internet connection or at least you should be on same Wi-Fi network. It is very convenient to do simple task like turn ON light or OFF light with our smart phone. You can build your own IoT home automation with this Experiment/Project. You may want to build a IoT project for college, school or university. In this we will see how to make ESP8266 web server which will control LED. We can control LED from any web browser on laptop, PC or Mobile phone by using same WiFi network. Once you learned this you will not only able to control LED you can literally control anything from light, fan, AC to big machines in factory. In this tutorial we will see how to make ESP8266 web server which will control LED. We can control LED from any web browser on laptop, PC or Mobile phone by using same WiFi network. Once you learned this you will not only able to control LED you can literally control anything from light, fan, AC to big machines in factory.

2. Components Required:

1. Node MCU ESP8266 12E,
2. 220 ohms Resistor,
3. LED,
4. Breadboard,
5. Jumper Wire.

3. Diagrammatic Connection of the components:



4. Arduino Library Required:

- ESP8266WiFi.h
- ESP8266WebServer

5. Arduino Code:

```
#include &lt;ESP8266WiFi.h&gt;
// Enter your wifi network name and Wifi Password
const char* ssid = &quot;Your SSID&quot;;
const char* password = &quot;Your Password&quot;;
// Set web server port number to 80
WiFiServer server(80);
// Variable to store the HTTP request
String header;
// These variables store current output state of LED
String outputRedState = &quot;off&quot;;
String outputGreenState = &quot;off&quot;;
String outputYellowState = &quot;off&quot;;
// Assign output variables to GPIO pins
const int redLED = 2;
const int greenLED = 4;
const int yellowLED = 5;
// 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(redLED, OUTPUT);
pinMode(greenLED, OUTPUT);
pinMode(yellowLED,OUTPUT);

// Set outputs to LOW
digitalWrite(redLED, LOW);
digitalWrite(greenLED, LOW);
digitalWrite(yellowLED, LOW);
// Connect to Wi-Fi network with SSID and password
Serial.print(&quot;Connecting to &quot;);
Serial.println(ssid);
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
```

```
delay(500);
Serial.print(&quot;.&quot;);
}
// Print local IP address and start web server
Serial.println(&quot;&quot;);
Serial.println(&quot;WiFi connected.&quot;);
Serial.println(&quot;IP address: &quot;);
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(&quot;New Client.&quot;); // print a message out in the serial
port
String currentLine = &quot;&quot;; // make a String to hold incoming data from
the client
currentTime = millis();
previousTime = currentTime;
while (client.connected() &amp;&amp; currentTime - previousTime &lt;=
timeoutTime) { // loop while the client&#39;s connected
currentTime = millis();
if (client.available()) { // if there&#39;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 == &#39;\n&#39;) { // if the byte is a newline character
// if the current line is blank, you got two newline characters in a
row.
// that&#39;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&#39;s coming, then a blank
line:
client.println(&quot;HTTP/1.1 200 OK&quot;);
client.println(&quot;Content-type:text/html&quot;);
client.println(&quot;Connection: close&quot;);
client.println();
```

```
// turns the GPIOs on and off
if (header.indexOf(&quot;GET /2/on&quot;) &gt;= 0) {
Serial.println(&quot;RED LED is on&quot;);
outputRedState = &quot;on&quot;;
digitalWrite(redLED, HIGH);
} else if (header.indexOf(&quot;GET /2/off&quot;) &gt;= 0) {
Serial.println(&quot;RED LED is off&quot;);
outputRedState = &quot;off&quot;;
digitalWrite(redLED, LOW);
} else if (header.indexOf(&quot;GET /4/on&quot;) &gt;= 0) {
Serial.println(&quot;Green LED is on&quot;);
outputGreenState = &quot;on&quot;;
digitalWrite(greenLED, HIGH);
} else if (header.indexOf(&quot;GET /4/off&quot;) &gt;= 0) {
Serial.println(&quot;Green LED is off&quot;);
outputGreenState = &quot;off&quot;;
digitalWrite(greenLED, LOW);
} else if (header.indexOf(&quot;GET /5/on&quot;) &gt;= 0) {
Serial.println(&quot;Yellow LED is on&quot;);
outputYellowState = &quot;on&quot;;
digitalWrite(yellowLED, HIGH);
} else if (header.indexOf(&quot;GET /5/off&quot;) &gt;= 0) {
Serial.println(&quot;Yellow LED is off&quot;);
outputYellowState = &quot;off&quot;;
digitalWrite(yellowLED, LOW);
}
// Display the HTML web page
client.println(&quot;&lt;!DOCTYPE html&gt;&lt;html&gt;&quot;);
client.println(&quot;&lt;head&gt;&lt;meta name=\&quot;viewport\&quot;
content=\&quot;width=device-
width, initial-scale=1\&quot;&gt;&quot;);
client.println(&quot;&lt;link rel=\&quot;icon\&quot;
href=\&quot;data:,\&quot;&gt;&quot;);
// CSS to style the on/off buttons
client.println(&quot;&lt;style&gt;html { font-family: Helvetica; display: inline-
block; margin: 0px auto; text-align: center;}&quot;);
client.println(&quot;.buttonRed { background-color: #ff0000; border: none;
color: white; padding: 16px 40px; border-radius: 60%;&quot;);
client.println(&quot;text-decoration: none; font-size: 30px; margin: 2px;
cursor: pointer;}&quot;);
```

```
client.println(&quot;.buttonGreen { background-color: #00ff00; border:
none; color: white; padding: 16px 40px; border-radius: 60%;&quot;);
client.println(&quot;text-decoration: none; font-size: 30px; margin: 2px;
cursor: pointer;}&quot;);
client.println(&quot;.buttonYellow { background-color: #feeb36; border:
none; color: white; padding: 16px 40px; border-radius: 60%;&quot;);
client.println(&quot;text-decoration: none; font-size: 30px; margin: 2px;
cursor: pointer;}&quot;);
client.println(&quot;.buttonOff { background-color: #77878A; border: none;
color: white; padding: 16px 40px; border-radius: 70%;&quot;);
client.println(&quot;text-decoration: none; font-size: 30px; margin: 2px;
cursor: pointer;}&lt;/style&gt;&lt;/head&gt;&quot;);

// Web Page Heading
client.println(&quot;&lt;body&gt;&lt;h1&gt;My LED Control
Server&lt;/h1&gt;&quot;);
// Display current state, and ON/OFF buttons for GPIO 2 Red LED
client.println(&quot;&lt;p&gt;Red LED is &quot; + outputRedState +
&quot;&lt;/p&gt;&quot;);
// If the outputRedState is off, it displays the OFF button
if (outputRedState==&quot;off&quot;) {
client.println(&quot;&lt;p&gt;&lt;a href=\&quot;/2/on\&quot;&gt;&lt;button
class=\&quot;button
buttonOff\&quot;&gt;OFF&lt;/button&gt;&lt;/a&gt;&lt;/p&gt;&quot;);
} else {
client.println(&quot;&lt;p&gt;&lt;a href=\&quot;/2/off\&quot;&gt;&lt;button
class=\&quot;button
buttonRed\&quot;&gt;ON&lt;/button&gt;&lt;/a&gt;&lt;/p&gt;&quot;);
}
// Display current state, and ON/OFF buttons for GPIO 4 Green LED
client.println(&quot;&lt;p&gt;Green LED is &quot; + outputGreenState +
&quot;&lt;/p&gt;&quot;);
// If the outputGreenState is off, it displays the OFF button
if (outputGreenState ==&quot;off&quot;) {
client.println(&quot;&lt;p&gt;&lt;a href=\&quot;/4/on\&quot;&gt;&lt;button
class=\&quot;button
buttonOff\&quot;&gt;OFF&lt;/button&gt;&lt;/a&gt;&lt;/p&gt;&quot;);
} else {
client.println(&quot;&lt;p&gt;&lt;a href=\&quot;/4/off\&quot;&gt;&lt;button
class=\&quot;button
```

```
buttonGreen\&quot;&gt;ON&lt;/button&gt;&lt;/a&gt;&lt;/p&gt;&quot;);
}
client.println(&quot;&lt;/body&gt;&lt;/html&gt;&quot;);
// Display current state, and ON/OFF buttons for GPIO 5 Yellow LED
client.println(&quot;&lt;p&gt;Yellow LED is &quot; + outputYellowState +
&quot;&lt;/p&gt;&quot;);
// If the outputYellowState is off, it displays the OFF button
if (outputYellowState ==&quot;off&quot;) {
client.println(&quot;&lt;p&gt;&lt;a href=\&quot;/5/on\&quot;&gt;&lt;button
class=\&quot;button
buttonOff\&quot;&gt;OFF&lt;/button&gt;&lt;/a&gt;&lt;/p&gt;&quot;);
} else {
client.println(&quot;&lt;p&gt;&lt;a href=\&quot;/5/off\&quot;&gt;&lt;button
class=\&quot;button
buttonYellow\&quot;&gt;ON&lt;/button&gt;&lt;/a&gt;&lt;/p&gt;&quot;);
}
client.println(&quot;&lt;/body&gt;&lt;/html&gt;&quot;);
// 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 = &quot;&quot;;
}
} else if (c != &#39;\r&#39;) { // 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 = &quot;&quot;;
// Close the connection
client.stop();
Serial.println(&quot;Client disconnected.&quot;);
Serial.println(&quot;&quot;);
}
}
```

6. Screenshot of the HTML page with unique IP addresses