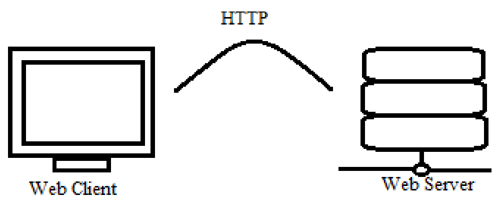
# Web Server using ESP8266.

**WHAT IS A WEB SERVER?**

A web server is a place that stores, processes and delivers web pages to clients(i.e. simple web browsers). Communication here in takes place via HTTP(Hyper Text Transfer Protocol). In this protocol, the client initiates communication by making a request for a specific web page using HTTP and the server responds with the content of that web page. Mostly it returns HTML documents.



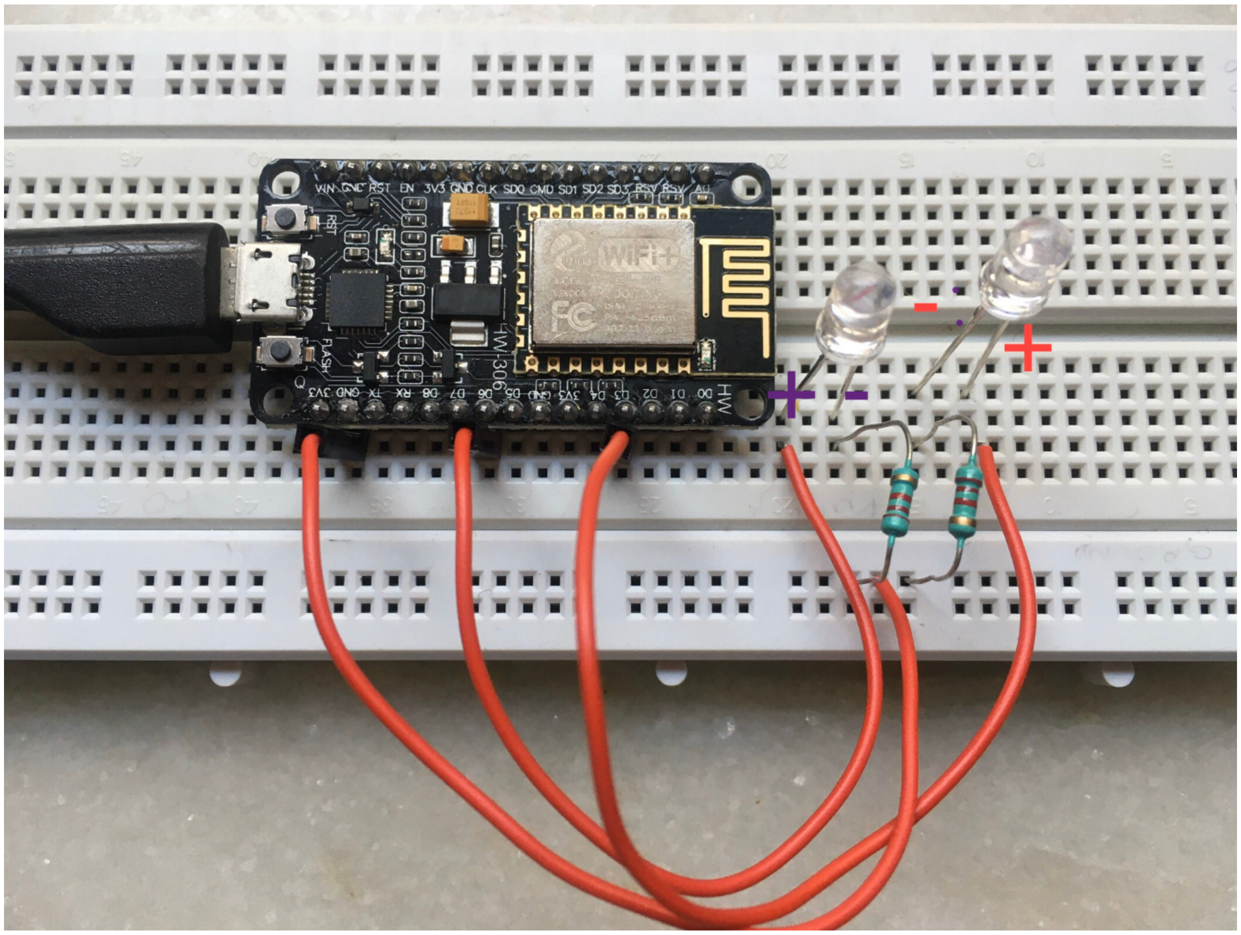
This what we shall do by using ESP8266. For example, we entered a URL like http://192.168.1.1/ledon in a browser, then it sends an HTTP request to ESP8266 to handle this request. ESP8266 then reads this request and turns the LED ON and sends a dynamic web page to the browser showing LED status : ON.

**MODES OF OPERATION:**

ESP8266 can not only connect to an existing WiFi network and act as a Web Server, but also set up a network of its own, allowing other devices to connect directly to it and access web pages. It can operate in three different modes viz. Station mode(STA), Soft Access Point mode(AP) and both STA and AP.
In **STA(Station mode)**, ESP8266 connects to an existing WiFi network. It gets IP from wireless router to which it is connected and sets up a web server to deliver web pages to all connected devices **under the existing WiFi network.**
In **Access Point (AP)** mode, ESP8266 creates its own WiFi network and acts as a hub (Just like WiFi router) for one or more stations, the medium is now wireless. In this mode, it creates a new WiFi network and sets SSID (Name of the network) and IP address to it and delivers web pages to all connected devices **under its own network**.

**WIRING:**

Connect LED1, LED2 to D3, D6 of ESP8266 module as shown below.



## ESP8266 as Web Server in Access Point (AP) mode:

The following libraries are essential so that ESP8266 can handle HTTP requests.

```
#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>
```

We also need to specify SSID, password, IP address, subnet mask and gateway.

```
const char* ssid = "Type_your_SSID";
const char* password = "Enter its password";

IPAddress local_ip(192,168,1,1);
IPAddress gateway(192,168,1,1);
IPAddress subnet(255,255,255,0);
```

Declare an object of ESP8266 web server library. The default port address is 80.

```
ESP8266WebServer server(80);
```

Declare the GPIO pins to which LEDs are connected and their initial state.

```
uint8_t LED1pin = D3;
bool LED1status = LOW;

uint8_t LED2pin = D6;
bool LED2status = LOW;
```

In the setup function, set led1, led2 as output pins.

```
Serial.begin(115200); // begin serial at 115200 baud
pinMode(LED1pin,OUTPUT);
pinMode(LED2pin,OUTPUT);
```

Set up a soft access point to establish a WiFi network by proving SSID, Password, IP address, subnet mask and gateway.

```
WiFi.softAP(ssid, password);
WiFi.softAPConfig(local_ip, gateway, subnet);
delay(100);
```

'on' method takes two parameters.  The first one is URL path and the second one is the name of function which we want to execute when that URL is hit.

```
server.on("/", handle_OnConnect);
server.on("/led1on", handle_led1on);
server.on("/led1off", handle_led1off);
server.on("/led2on", handle_led2on);
server.on("/led2off", handle_led2off);
server.onNotFound(handle_NotFound);
```

The begin method is called to the start server.

```
server.begin();
Serial.println("HTTP server started");
```

handleClient() method handles incoming HTTP requests from client,

```
void loop() {
  server.handleClient();
  if(LED1status)
  {digitalWrite(LED1pin, HIGH);}
  else
  {digitalWrite(LED1pin, LOW);}

  if(LED2status)
  {digitalWrite(LED2pin, HIGH);}
  else
  {digitalWrite(LED2pin, LOW);}
}
```

In order to respond to the HTTP request, send method is used. It has three parameters: status code 200 which corresponds to 'OK'; content type as 'text/html' and SendHTML() custom function which creates a dynamic HTML page containing status of

LEDs.

```
void handle_OnConnect()
{
  LED1status = LOW;
  LED2status = LOW;
  Serial.println("D3 Status: OFF | D6 Status: OFF");
  server.send(200, "text/html", SendHTML(LED1status,LED2status));
}
```

To handle LED ON and OFF conditions.

```
void handle_led1on() {
  LED1status = HIGH;
  Serial.println("D3 Status: ON");
  server.send(200, "text/html", SendHTML(true,LED2status));
}

void handle_led1off() {
  LED1status = LOW;
  Serial.println(" D3 Status: OFF");
  server.send(200, "text/html", SendHTML(false,LED2status));
}

void handle_led2on() {
  LED2status = HIGH;
  Serial.println(" D6 Status: ON");
  server.send(200, "text/html", SendHTML(LED1status,true));
}

void handle_led2off() {
  LED2status = LOW;
  Serial.println("D6 Status: OFF");
  server.send(200, "text/html", SendHTML(LED1status,false));
}

void handle_NotFound(){
  server.send(404, "text/plain", "Not found");
}
```

SendHTML() function generates a web page whenever the web server gets a request from a web client.

```
String SendHTML(uint8_t led1stat,uint8_t led2stat){
String ptr = "<!DOCTYPE html> <html>\n";
```

<meta> viewport element makes the web page responsive in any web browser and title tag sets the title of the page.

```
ptr +="<head><meta name=\"viewport\" content=\"width=device-width, initial-scale=1.0, u
ptr +="<title>LED Control</title>\n";
```

Web Page is styled as below:

```
ptr +="<style>html { font-family: Helvetica; display: inline-block; margin: 0px auto;
ptr +="body{margin-top: 50px;} h1 {color: #444444;margin: 50px auto 30px;} h3 {color:
ptr +="p {font-size: 14px;color: #888;margin-bottom: 10px;}\n"
ptr +=".button {display: block;width: 80px;background-color: #1abc9c;border: none;colo
ptr +=".button-on {background-color: #1abc9c;}\n";
ptr +=".button-on:active {background-color: #16a085;}\n";
ptr +=".button-off {background-color: #34495e;}\n";
ptr +=".button-off:active {background-color: #2c3e50;}\n";
```

Heading is set as following:

```
ptr +="<h1>ESP8266 Web Server</h1>\n";
ptr +="<h3>Using Access Point(AP) Mode</h3>\n";
```

ON/OFF button is displayed as per the status of LED.

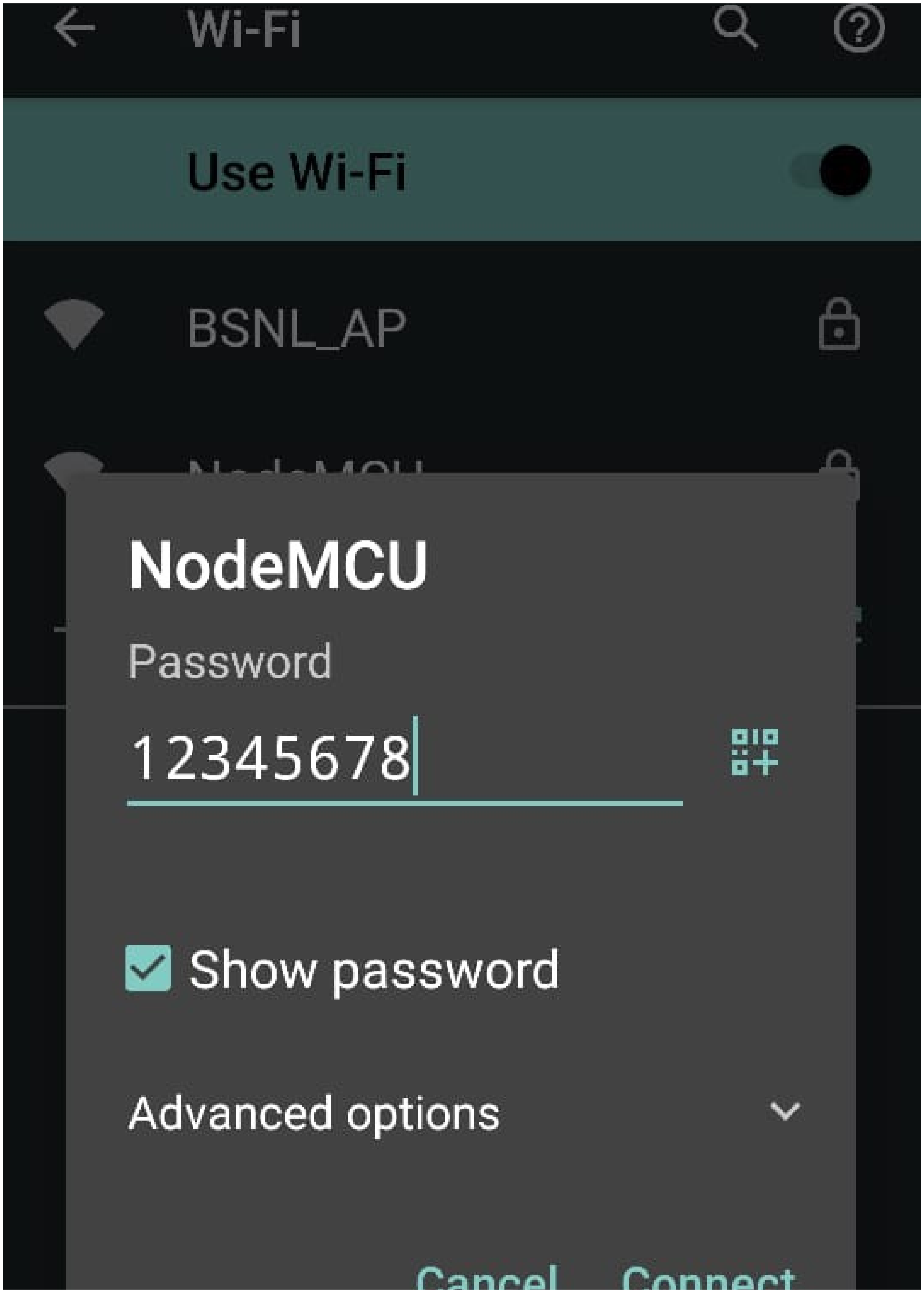
```
if(led1stat)
  {ptr +="<p>LED1 Status: ON</p><a class=\"button button-off\" href=\"/led1off\">OFF</
else
  {ptr +="<p>LED1 Status: OFF</p><a class=\"button button-on\" href=\"/led1on\">ON</a>

if(led2stat)
  {ptr +="<p>LED2 Status: ON</p><a class=\"button button-off\" href=\"/led2off\">OFF</
else
  {ptr +="<p>LED2 Status: OFF</p><a class=\"button button-on\" href=\"/led2on\">ON</a>
```

Final Code:

```

#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>

/* Put your SSID & Password */
const char* ssid = "NodeMCU";  // Enter SSID here
const char* password = "12345678";  //Enter Password here

/* Put IP Address details */
IPAddress local_ip(192,168,1,1);
IPAddress gateway(192,168,1,1);
IPAddress subnet(255,255,255,0);

ESP8266WebServer server(80);

uint8_t LED1pin = D3;
bool LED1status = LOW;

uint8_t LED2pin = D6;
bool LED2status = LOW;

void setup() {
  Serial.begin(115200);
  pinMode(LED1pin, OUTPUT);
  pinMode(LED2pin, OUTPUT);
  WiFi.softAP(ssid, password);
  WiFi.softAPConfig(local_ip, gateway, subnet);
  delay(100);

  server.on("/", handle_OnConnect);
  server.on("/led1on", handle_led1on);
  server.on("/led1off", handle_led1off);
  server.on("/led2on", handle_led2on);
  server.on("/led2off", handle_led2off);
  server.onNotFound(handle_NotFound);

  server.begin();
  Serial.println("HTTP server started");
}
void loop() {
  server.handleClient();
  if(LED1status)
  {digitalWrite(LED1pin, HIGH);}
  else
  {digitalWrite(LED1pin, LOW);}

  if(LED2status)
  {digitalWrite(LED2pin, HIGH);}
  else
  {digitalWrite(LED2pin, LOW);}
}

void handle_OnConnect() {
  LED1status = LOW;
  LED2status = LOW;
  Serial.println("D3 Status: OFF | D6 Status: OFF");
  server.send(200, "text/html", SendHTML(LED1status,LED2status));
}

void handle_led1on() {
  LED1status = HIGH;
  Serial.println(" D3 Status: ON");
  server.send(200, "text/html", SendHTML(true,LED2status));
}

void handle_led1off() {
  LED1status = LOW;
  Serial.println("D3 Status: OFF");
  server.send(200, "text/html", SendHTML(false,LED2status));
}

void handle_led2on() {
  LED2status = HIGH;
  Serial.println(" D6 Status: ON");
  server.send(200, "text/html", SendHTML(LED1status,true));
}

void handle_led2off() {
  LED2status = LOW;
  Serial.println("D6 Status: OFF");
  server.send(200, "text/html", SendHTML(LED1status,false));
}

void handle_NotFound(){
  server.send(404, "text/plain", "Not found");
}

String SendHTML(uint8_t led1stat,uint8_t led2stat){
  String ptr = "<!DOCTYPE html> <html>\n";
  ptr +="<head><meta name=\"viewport\" content=\"width=device-width, initial-scale=1.0,
  ptr +="<title>LED Control</title>\n";
  ptr +="<style>html { font-family: Helvetica; display: inline-block; margin: 0px auto
  ptr +="body{margin-top: 50px;} h1 {color: #444444;margin: 50px auto 30px;} h3 {color:
  ptr +=".button {display: block;width: 80px;background-color: #1abc9c;border: none;col
  ptr +=".button-on {background-color: #1abc9c;}\n";
  ptr +=".button-on:active {background-color: #16a085;}\n";
  ptr +=".button-off {background-color: #34495e;}\n";
  ptr +=".button-off:active {background-color: #2c3e50;}\n";
  ptr +="p {font-size: 14px;color: #888;margin-bottom: 10px;}\n";
  ptr +="</style>\n";
  ptr +="</head>\n";
  ptr +="<body>\n";
  ptr +="<h1>ESP8266 Web Server</h1>\n";
  ptr +="<h3>Using Access Point(AP) Mode</h3>\n";

   if(led1stat)
  {ptr +="<p>LED1 Status: ON</p><a class=\"button button-off\" href=\"/led1off\">OFF</a
  else
  {ptr +="<p>LED1 Status: OFF</p><a class=\"button button-on\" href=\"/led1on\">ON</a>\

  if(led2stat)
  {ptr +="<p>LED2 Status: ON</p><a class=\"button button-off\" href=\"/led2off\">OFF</a
  else
  {ptr +="<p>LED2 Status: OFF</p><a class=\"button button-on\" href=\"/led2on\">ON</a>\

  ptr +="</body>\n";
  ptr +="</html>\n";
  return ptr;
}
```

**FINAL STEP:**

1.Upload the sketch and open the Serial Monitor at a baud rate of 115200. Also press the RESET button on ESP8266. 'HTTP' server started' will be then displayed on the screen.
2.On your phone or laptop look for ESP8266's network name and connect to it.



Wi-Fi

Use Wi-Fi

BSNL_AP

NodeMCU

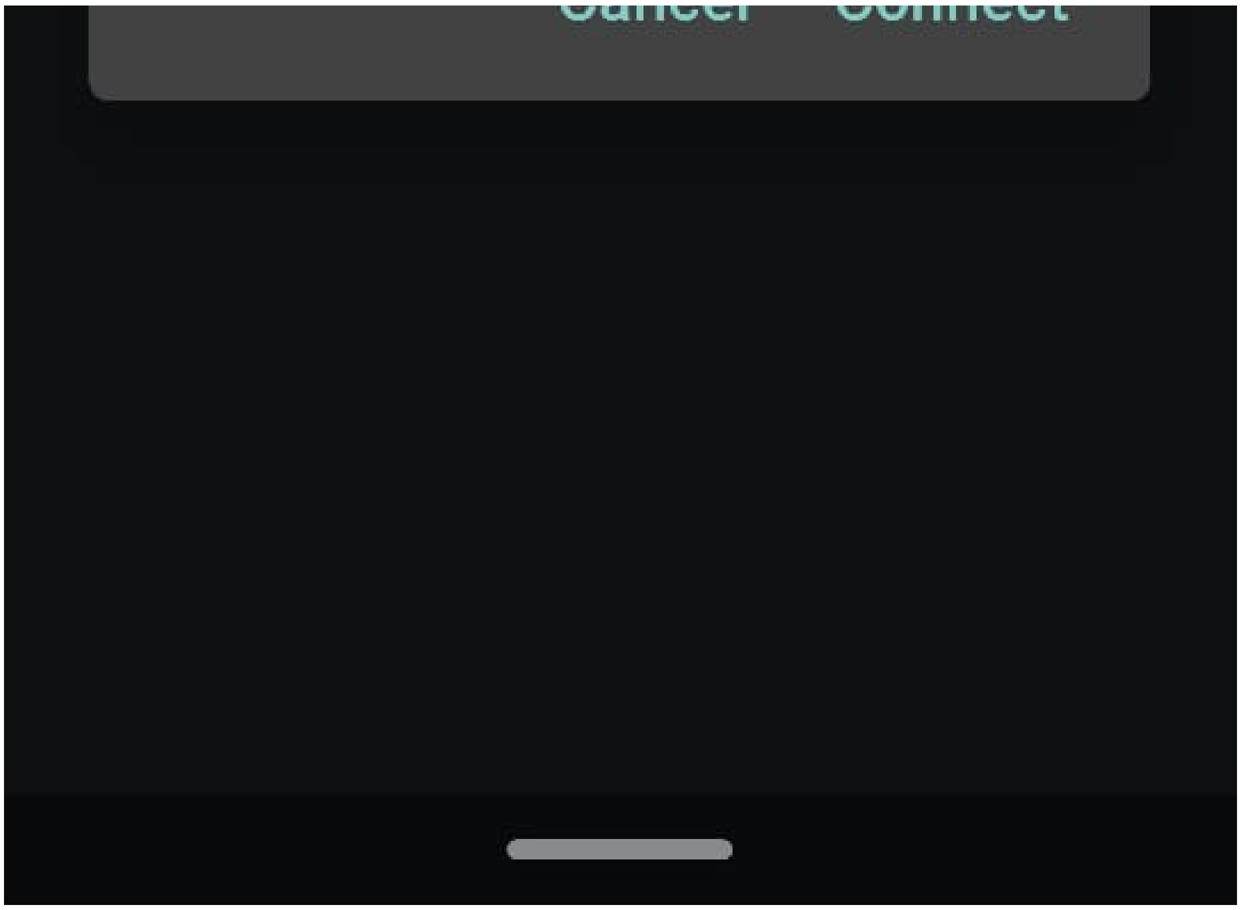
Password

12345678

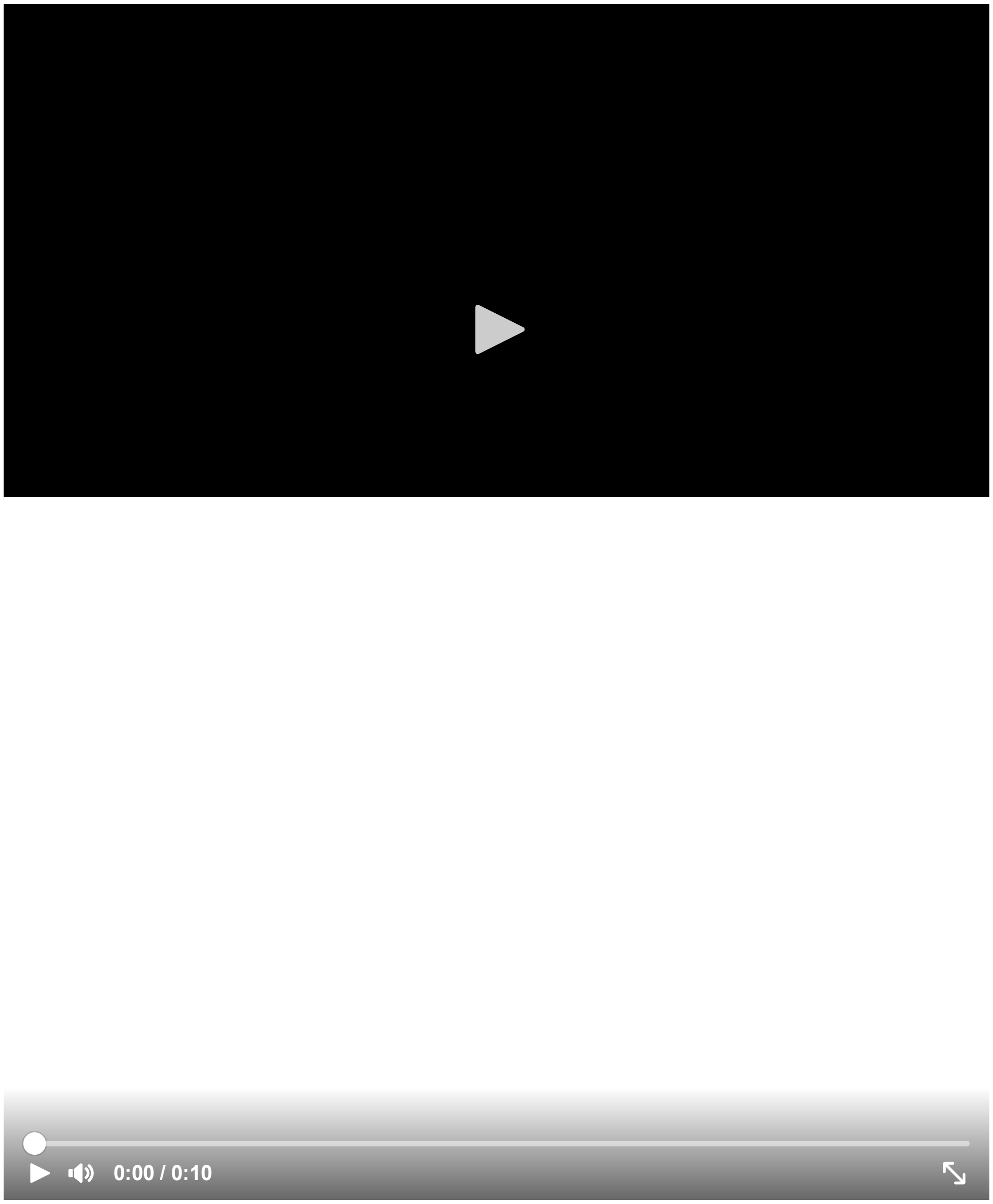
Show password [✓]

Advanced options

Cancel Connect



3.Then point your browser to 192.168.1.1
4.A  web page appears with the current status of LEDs and two buttons to control them.
5.Click the button to turn LED1 ON. ESP8266 receives a request for /led1on URL.
6.It then turns the LED1 ON updates LED status on the page. Similarly test LED2.


0:00 / 0:10