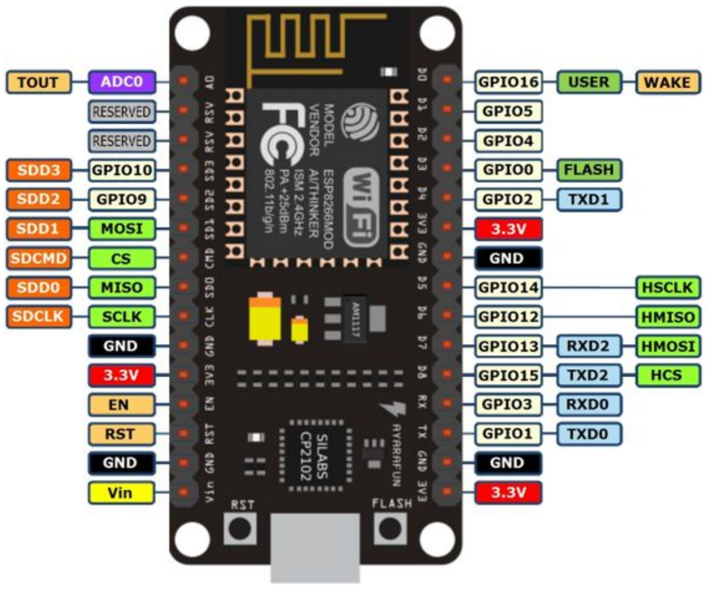
IoT Using ESP8266 (Control LED)

Having WiFi ESP8266 opens door for us to make interesting IoT stuff on that can have practical value in our lifestyle. Many smart home automation projects have been and are being made with the help of WiFi or Bluetooth using ESP8266 or other resembling micro-controllers. Open source documentation and a large helping community play a major role in bringing these things possible for every low tech person to develop smart IoT based systems for making their lifestyle easier, efficient and smart.

Pinout of ESP8266: IoT means automating your services/works through internet and in our case automate electronics based project. For this we need input/output devices to be connected with ESP8266. Input/output can be as sensors (e.g. distance sensor) and actuators (e.g. motors and led) respectively. ESP8266 offers multiple digital pins and one analog pin, other details of the pins can be seen in below figure by Instructables. For this tutorial we won't go into the details of all pins.



Simple Blinking LED: We will start with simple LED blink program to keep things step by step and slow so that no difficulty should come once we move to advanced things.

Code:

```
void setup() {
// put your setup code here, to run once:
pinMode(2,OUTPUT); // 2 pin of ESP8266 is active low that means it will switch on low voltage and vice versa.
}
void loop() {
// put your main code here, to run repeatedly:
digitalWrite(2,0); // Turn on LED
delay(1000);
digitalWrite(2,1); // Turn off LED
delay(1000);
}
```

Code Explanation:

- Setting pin 2 as output in void function using **pinMode(pin,mode)** function.
- Turn on LED on pin 2 (active low) for 1 second.
- Turn off LED on pin 2 (active low) for 1 second.

Output: Built-in LED on ESP8266 should start blinking.

Controlling LED using WiFi: Now let's jump to some basic IoT stuff by controlling LED through a web page. For that we will have to do following.

- Give ESP8266 a WiFi connection.
- Create and start server with ESP8266.
- Wait for client (any device or computer) to connect.
- Takes request from client (Turn on/off LED in our case).
- Act accordingly to the request (Turn on/off LED in our case).
- Create HTML page and display the inputs so that client (we) can request server (ESP8266) to entertain (control LED) our request.

Code:

```
#include <ESP8266WiFi.h>
// Replace with your network credentials
String ssid = "CRAIB-LAB";
String password = "********";
// Create an instance of the server on port 80
WiFiServer server(80);
void setup() {
Serial.begin(115200);
pinMode(2, OUTPUT); // Set GPIO2 as an output pin
digitalWrite(2, HIGH); // Turn off the LED initially
// Connect to Wi-Fi network
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
delay(1000);
Serial.println("Connecting to WiFi...");
}
// Start the server
server.begin();
Serial.println("Server started");
Serial.print("IP address:\t");
Serial.println(WiFi.localIP());
}
void loop() {
// Wait for a client to connect
WiFiClient client = server.available();
if (!client) {
return;
}
// Wait for a request from the client
while(!client.available()){
delay(1);
}
// Read the first line of the request
String request = client.readStringUntil('\r');
Serial.println(request);
client.flush();
// If the request is to turn on the LED, turn it on
if (request.indexOf("/LED=ON") != -1) {
digitalWrite(2, 0);
}
// If the request is to turn off the LED, turn it off
if (request.indexOf("/LED=OFF") != -1) {
digitalWrite(2, 1);
}
// Send the response back to the client
client.println("HTTP/1.1 200 OK");
client.println("Content-Type: text/html");
client.println("");
client.println("<!DOCTYPE HTML>");
client.println("<html>");
client.println("<body>");
client.println("<h1>ESP8266 LED Control</h1>");
client.println("<p>Click <a href=\"/LED=ON\">here</a> to turn the LED on</p>");
client.println("<p>Click <a href=\"/LED=OFF\">here</a> to turn the LED off</p>");
client.println("</body>");
client.println("</html>");

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

Code Explanation:

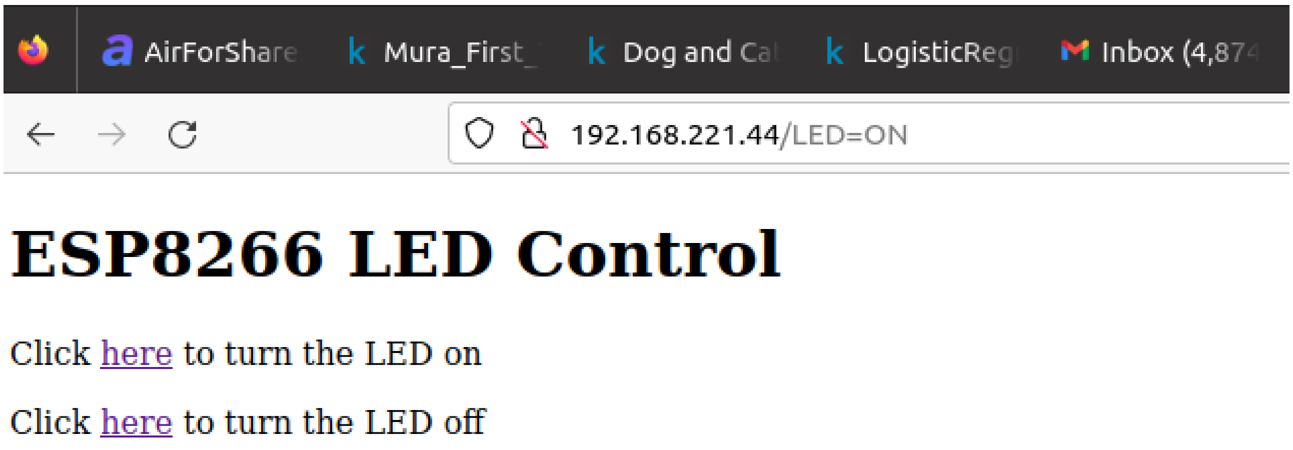
- Include ESP8266 WiFi library.
- Provide credentials of WiFi to let ESP8266 connect to your network.
- Start ESP8266 server on port number 80.
- Set the baud rate (115200), define the mode of pin 2 (output) and turn it off initially.
- Connect the WiFi using for given WiFi credentials and checking its connectivity over each second.
- Start the sever using **begin()** function and print the IP on serial monitor to make any connect client.
- Wait for a client (we with any device by use of IP on serial monitor) to connect and wait for its request (Turn on/off LED in our case).

- Read the request, store in a string and print it on serial monitor.
- Check the status of client's request to automate LED accordingly.
- Create the web page for client to send the request and act accordingly.

Output: You can see below after taking time to connect with WiFi ESP8266's server has started and IP address is also printed on serial monitor so that any client can use it connect to the server.

```
Connecting to WiFi...
Connecting to WiFi...
Connecting to WiFi...
Connecting to WiFi...
Connecting to WiFi...
Server started
IP address:     192.168.221.44
```

HTML web page for client: Type this IP address on browser with any client (computer or any other device) and it will get you on following page. Send requests on underlined here to turn on/off LED and it will act accordingly.



ESP8266 LED Control

Click here to turn the LED on

Click here to turn the LED off

Notification on serial monitor: Once both server and client are ready to communicate, upon every request notification can be monitored on serial monitor as shown below.

```
GET / HTTP/1.1
Client disconnected
GET /favicon.ico HTTP/1.1
Client disconnected
GET /LED=OFF HTTP/1.1
Client disconnected
GET /LED=ON HTTP/1.1
Client disconnected
GET /LED=OFF HTTP/1.1
Client disconnected
```