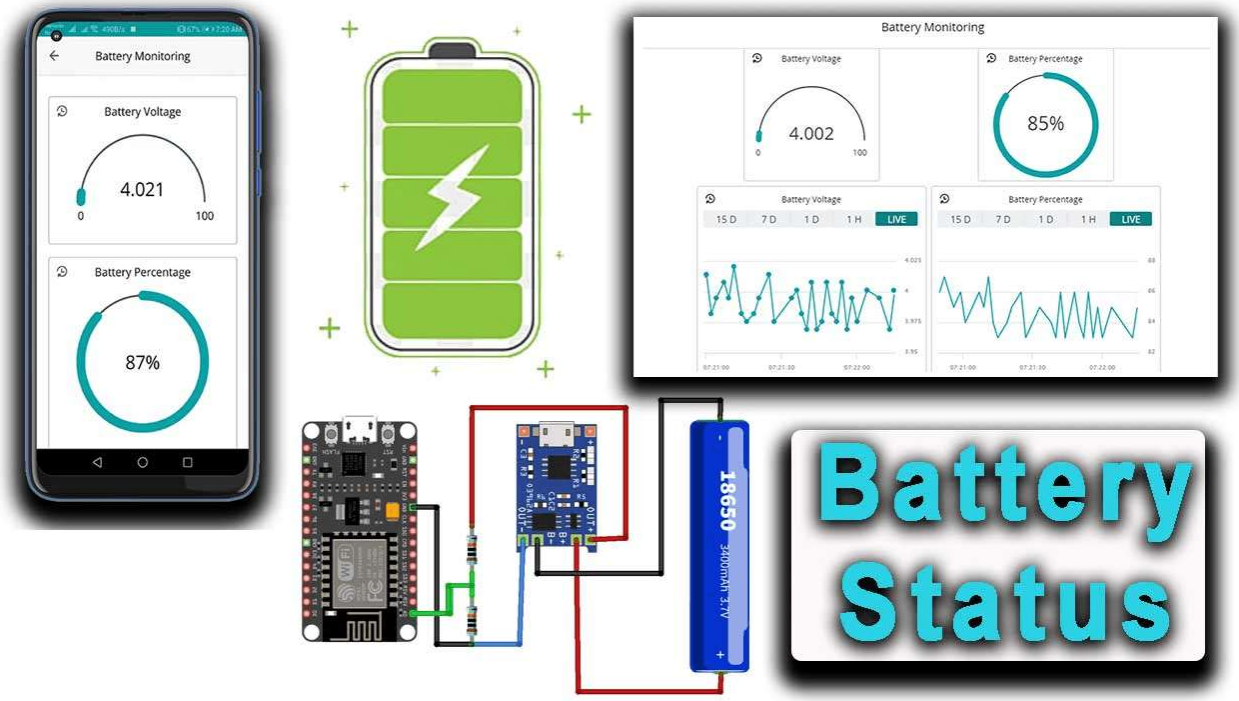


Send data to web server using ESP8266 (Esp8266 HTTP requests)

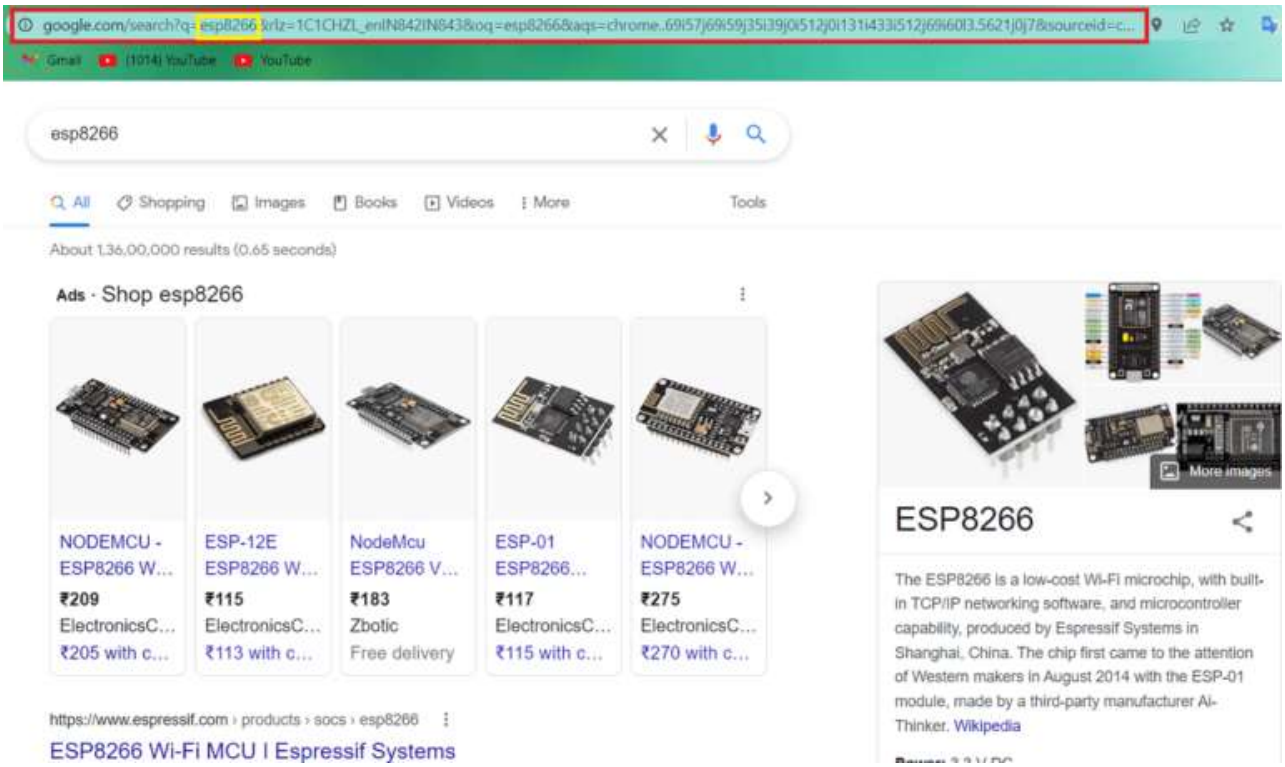
Leave a Comment / Internet of Things (IoT) / By Bhimsen

In the previous two IoT articles, I have introduced you to the esp8266 module and the AT commands also. Today I will explain how you can send data to a web server using ESP8266. In other words, you can say how you can make an HTTP request using ESP8266. For those who don't know what an HTTP request is, let me explain.

Table of Contents

Introduction to the HTTP request

HTTP is a protocol to transfer the data to and fro the client and server. Here the client is your browser and the server is the remote (cloud) storage where a website is hosted. To access the files of the server we make an HTTP request with the address of that file. For example, if you search anything on google then your browser will embed your query and some other information about your browser in a link with the server location of google. This process happens in the background but you can see the link that your browser is requesting in the URL bar of the browser.



You can see the search text in the URL. This is how they send any parameter in the URL. Here you can also see that our search term equates to the q parameter and many other parameters are there in the URL. They contain information about the client (browser) and are separated with &. That means the server is listening to these parameters in the URL. As the parameter matches, the server catches and stores the value of that parameter. If you didn't get it now, don't worry. I will explain this practically by creating my own server in localhost (computer).

Types of HTTP requests

There are two types of HTTP requests. The first is the get method and the second is the post method.

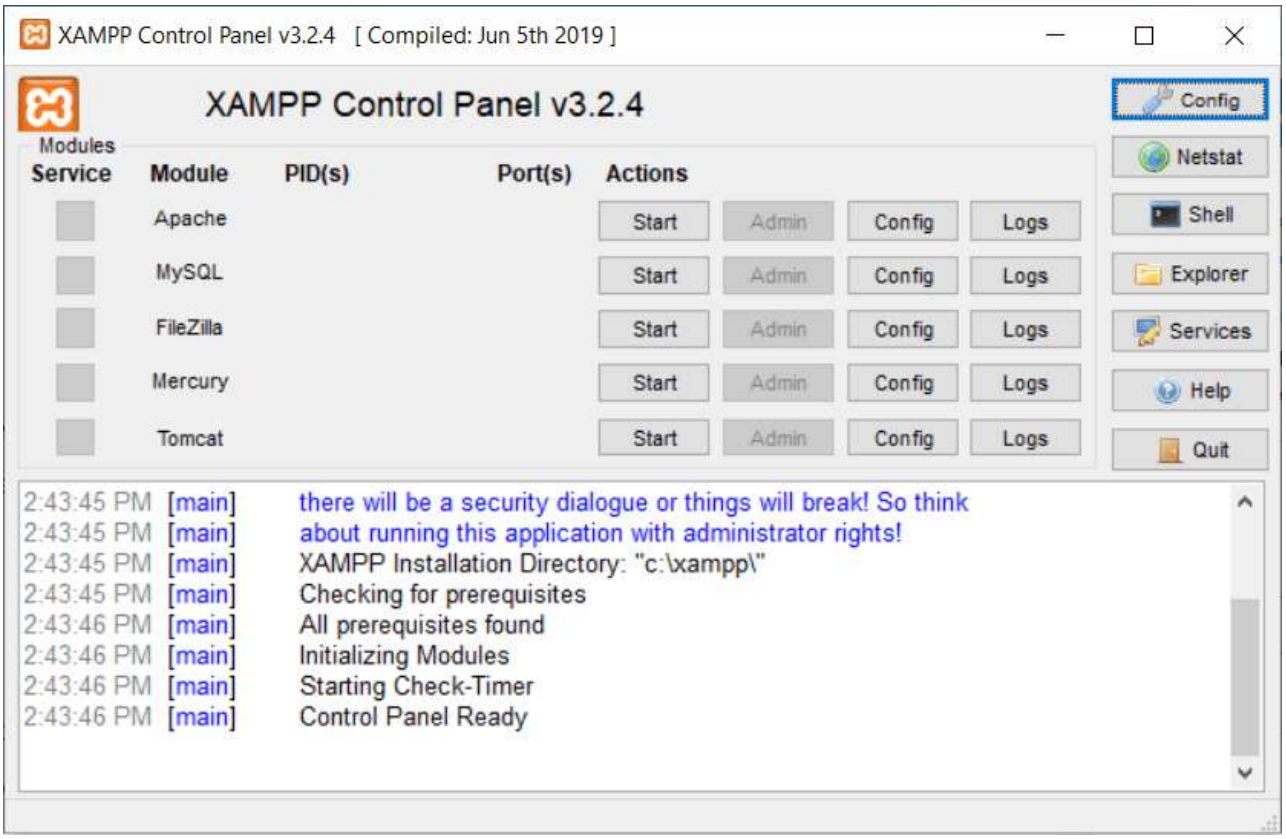
In the get method, parameters are embedded in the URL and can be seen. As we have seen in google's example. We send non-sensitive data to the server using this method. We cannot send binary data using this method like images, songs, videos, etc.

In the post method, parameters are not shown in the URL. Data transfer happens in an encrypted way in the background. Using this method we transfer sensitive data like passwords. We can send the binary data also.

Creating the web server on the localhost

To demonstrate to you I will create a server on my computer. To create the server on the localhost we need some tools that allow our computer to serve files to the browser as a server. The tool (software) I will be using is the XAMPP.

You need to download and install the XAMPP on the computer. Click here to download. This software enables the apache web server on your computer. Through this tool, you can enable an apache web server, MySQL database, FileZilla (to upload files to the webserver), and many more things. The XAMPP control panel looks like this.



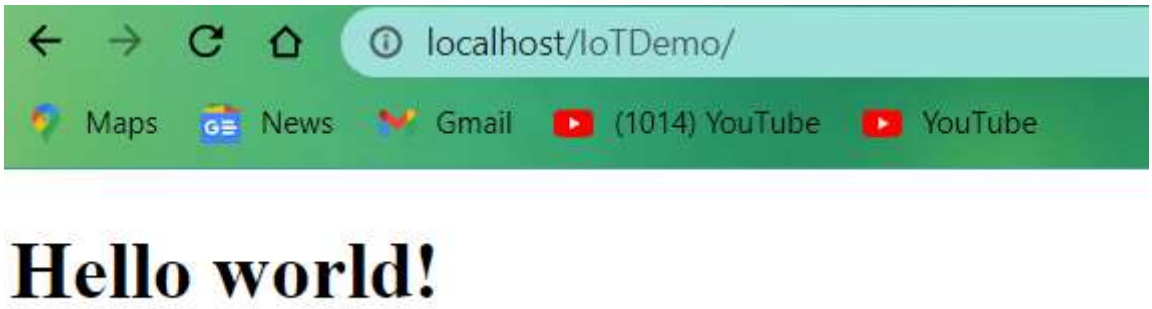
To create the server you need to just start the Apache in the control panel. Now the server is running on our computer. To access the server you need to type localhost in the browser URL section. You will be redirected to the dashboard of the server. To create a directory (website) you need to search the file named htdocs under the XAMPP installation directory. You can simply search in the start menu of your computer.

Make a directory through which you want to access your website (just like the domain name). I will make the directory with the name “IoTDemo”. Now I can access the files under this directory using localhost/IoTDemo.

To make this look like a webpage you need to add some HTML files under this directory. So create a file with the name “index.html”. Now the reason why we generally use the home page with the name index is that server treats it primarily as a home page. Until you don’t specify the home page in the .htaccess file. Now add this code to the file.

```
<!DOCTYPE html>
<head>
    <title>Hello world!</title>
</head>
<body>
    <h1>Hello world!</h1>
</body>
</html>
```

Now your page will look like this.



Creating HTTP request handler

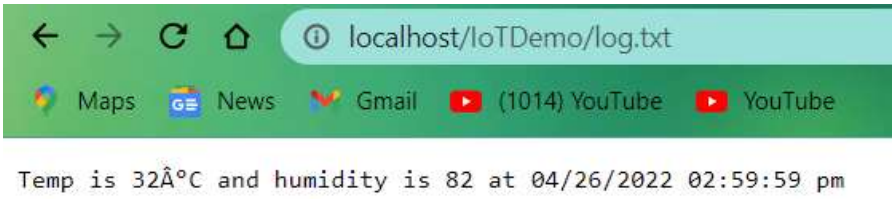
To handle the HTTP request we have to create a handler on our server. I will make an HTTP request handler in the PHP programming language. So create a file “handler.php” and add the following code in the file.

```
<?php
$Temp = $_GET['temp'];                  //get the value of temp variable fr
$Humidity = $_GET['hum'];               //get the value of hum variable frc
$date = date('m/d/Y h:i:s a', time());  //current date time

//Embedd the temperature and humidity data along with date time in a string
$fileContent = "Temp is ".$Temp."°C and humidity is ".$Humidity." at ".$dat

//Append the created string into a text file.
$status = file_put_contents('log.txt',$fileContent,FILE_APPEND);

//File writing status
if($status)
    echo "Data written successfully.";
else
    echo "Something went wrong!";
?>
```

In this code, I have listened to two parameters in the HTTP URL. The first is the temp variable and the second is the hum variable. The third variable I have used to store the current date and time. This data will not be in the URL instead I have used server date and time. This temperature and humidity data will be concatenated along with the date and time. Then I have appended the final string into a text file. I made the text file beforehand called 'log.txt'.

Now I can request the URL from the browser and log the temp and humidity data along with the time on the server. The URL looks like this "localhost/IoTDemo/handler.php?temp=32&hum=82". Where the temperature (temp) value is 32 and the humidity (hum) value is 82. Now let's see the practical approach.

Now let's see whether the data is actually written in the file or not.



As you can see that data actually gets written in the file.

Make HTTP request using ESP8266 (send data to web server using esp8266)

You have seen how you can send data to web server using a browser but now I will explain how you can send data to web server using esp8266. I will explain to you two different methods to make an HTTP request. The first is using AT commands and second is the programming the esp8266 board. I will also explain how to make both types of HTTP requests using esp8266. So first I will explain the AT command method.

To send the AT commands to the ESP8266 you have to first connect ESP8266 to a USB to serial converter because ESP8266 uses UART to communicate to the other devices. Second, you have to use a tool that sends serial data over any port of computer via USB. I'm using a software called Realterm. Click here to go to the download page.

AT command to make get requests in esp8266

To make an HTTP request using an esp8266 through the AT command we have to first connect to the server using this command.

> *AT+CIPSTART="TCP","electronics-fun.com",80\r\n*

Here electronics fun is the server and 80 is the HTTP port and connection type is TCP. then we have to send these commands.

> *AT+CIPSEND=71\r\n*
> *GET /handler.php?temp=32&hum=82 HTTP/1.1\r\nHost: electronics-fun.com\r\n\r\n*

Here 71 is the length of the URL including carriage(\r) return and newline(\n). But first, make sure that your esp8266 is connected to a network and is in station mode. You can see my previous IoT article for reference.

AT command to make post requests in esp8266

To make an HTTP post request using an esp8266 through the AT command we have to send these commands to the esp8266. Here the server connection part is common for both types of requests.

> *AT+CIPSEND=142\r\n*

After sending this command ESP8266 will ask you to enter a string of 142 characters long. Then you have o enter the following string.

> *POST /receiver.php HTTP/1.1\r\nHost: electronics-fun.com\r\nContent-Type: application/x-www-form-urlencoded\r\nContent-Length: 11\r\n\r\ntemp=32&hum=80\r\n*

Now let's see how we can program esp8266 to make HTTP requests.

Program esp8266 to make get requests

Here is the code to program esp8266 to make a get request.

```
#include <ESP8266WiFi.h>

const char* ssid     = "WIFI_SSID";
const char* password = "WIFI_PASS";

const char* host = "electronics-fun.com"; // only electronics-fun.com not h

void setup() {
  Serial.begin(115200);
  delay(10);

  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");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}

void loop() {
  delay(5000);

  Serial.print("connecting to ");
  Serial.println(host);

  // Use WiFiClient class to create TCP connections
  WiFiClientSecure client;
  const int httpPort = 80; // 80 is for HTTP / 443 is for HTTPS!

  client.setInsecure(); // this is the magical line that makes everything w

  if (!client.connect(host, httpPort)) { //works!
    Serial.println("connection failed");
    return;
  }

  // We now create a URI for the request
  String url = "/handler.php?temp=30&hum=80";


  Serial.print("Requesting URL: ");
  Serial.println(url);

  // This will send the request to the server
  client.print(String("GET ") + url + " HTTP/1.1\r\n" +
               "Host: " + host + "\r\n" +
               "Connection: close\r\n\r\n");

  Serial.println();
```

```
  Serial.println("closing connection");
}
```

Program esp8266 to make post requests

Here is the code to program esp8266 to make a post request.

```
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>
#include <WiFiClient.h>

const char* ssid     = "WIFI_SSID";
const char* password = "WIFI_PASS";

const char* serverName = "http://electronics-fun.com/handler.php";

void setup() {
  Serial.begin(115200);

  WiFi.begin(ssid, password);
  Serial.println("Connecting");
  while(WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.print("Connected to WiFi network with IP Address: ");
  Serial.println(WiFi.localIP());
}

void loop() {
    //Check WiFi connection status
    if(WiFi.status()== WL_CONNECTED){
      WiFiClient client;
      HTTPClient http;

      // Your Domain name with URL path or IP address with path
      http.begin(client, serverName);

      // Specify content-type header
      http.addHeader("Content-Type", "application/x-www-form-urlencoded");
```

```
      // Data to send with HTTP POST
      String httpRequestData = "temp=32&hum=80";
      // Send HTTP POST request
      int httpResponseCode = http.POST(httpRequestData);

      Serial.print("HTTP Response code: ");
      Serial.println(httpResponseCode);

      // Free resources
      http.end();
    }
    else {
      Serial.println("WiFi Disconnected");
    }
    delay(1000);
}
```

So this is the code that you can upload to the esp8266 board to make a post request to a server.

← Previous Post Next Post →

Leave a Comment

Your email address will not be published. Required fields are marked *

Type here..

Name*

Email*

Website

Save my name, email, and website in this browser for the next time I comment.

POST COMMENT »

Search …

Recent Posts

Resistor Color code Calculator

LED resistor calculator

Comparison of Arduino boards

OpAmp gain calculator

Affiliate disclosure

ElectronicsFun is a participant in the Amazon Services LLC Associates Program an affiliate advertising program designed to provide a means for website owners to earn advertising fees by advertising and linking to amazon.com, audible.com, and any other website that may be affiliated

with Amazon Service LLC Associates Program. If you got any questions about this, feel free to contact us.

Categories

Arduino Programming Tutorial

Arduino Projects with Code

Basic Integrated Circuits (ICs)

Basic projects

Calculators and tools

Digital Electronics

Discrete Semiconductor

Electronics Projects

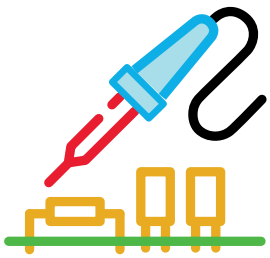
Electronics tools and equipment

Internet of Things (IoT)

Passive Components

Power Electronics

Electronics fun



about us

connect

About

Privacy Policy

Contact

Terms and Conditions