ESP32 Asynchronous Web Server – jQuery AJAX

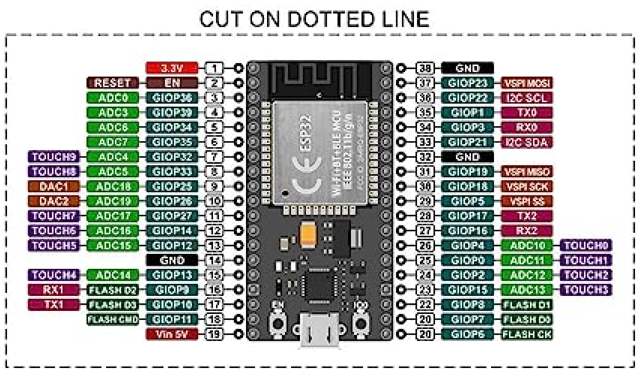
AJAX makes it possible to exchange data with the server and to update certain parts of the website without reloading the entire page. The jQuery ajax() method provides the basic functionality of AJAX in jQuery. This article shows how to use jQuery ajax() and ESPAsyncWebServer.

In this example we will use the code from the Esp32 Web Updater and SPIFFS file manager page. The following compressed folder contains all the files included in this article.

: ESP32 ASYNCWEBSERVER JQUERY AJAX PROJECT FILES

ESP32 PINOUT

Warm Tips: You can print this pin map in equal proportions and attach it to the top of the module to recognize the definition of each pin.

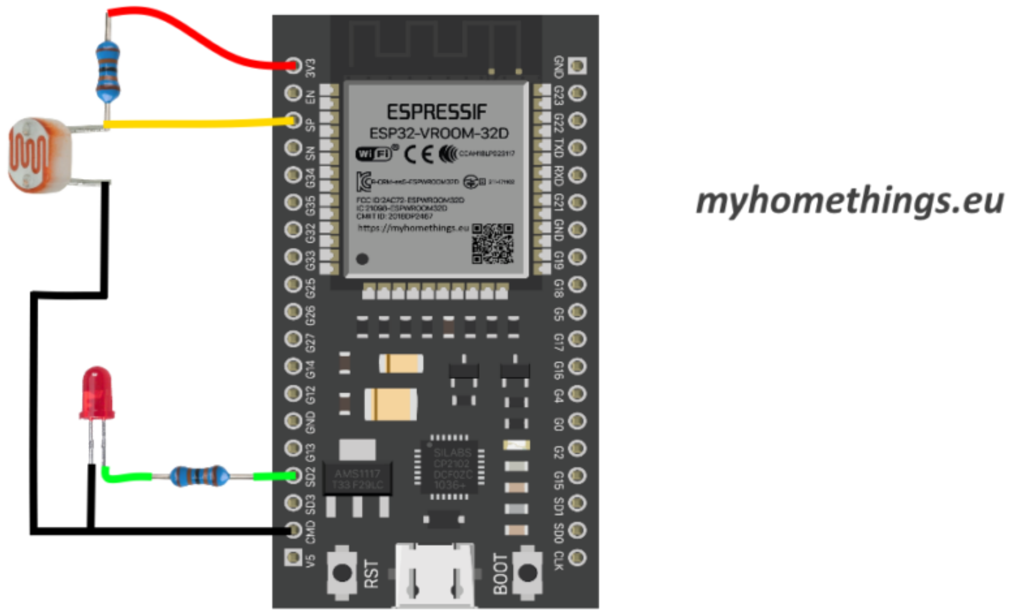


[ESP32 Development Board 2.4 GHz Dual Core WLAN WiFi + Bluetooth]()

advertising

[More detailed information on using the ESP32 Asynchronous web server here.]()

Let's prepare the hardware. Connect an LED with a 330 ohm resistor to the gpio2 pin of an ESP32 and connect a voltage divider consisting of an LDR and a 10k resistor to the gpio36 pin based on the image below.



Connecting LDR light sensor and LED to ESP32

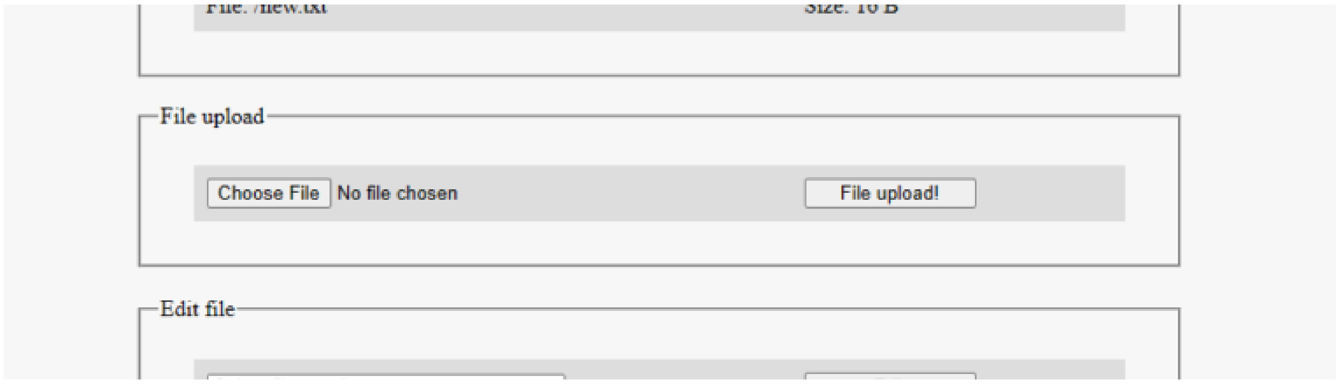
Unzip the downloaded compressed folder, then copy the Esp32-AsyncWebserver-jQuery-ajax folder to the .../Arduno directory. Open the Arduino IDE, select the file Esp32-AsyncWebserver-jQuery-ajax in the File – Sketchbook menu, enter the SSID and password required for the Wifi connection in the config.h file, then upload it to the ESP32 development board .

config.h

```
const char* ssid = "SSID";
const char* password = "Wifi_pessword";

const char* http_username = "admin";
const char* http_password = "admin";

const char* host = "esp32-asyncwebserver";

...
```

Open the address http://esp32-asyncwebserver.local/manager in the browser. The default login credentials are admin/admin. We can also change this in the config.h file.

On the page that opens, upload the jquery-3.6.3.min.js and index.html files from the project folder, then go to http://esp32-asyncwebserver.local.



Upload jquery-3.6.3.min.js and index.html files to ESP32

In the config.h file, enter the name of the uploaded jQuery file with a slash at the beginning. The variables of the code snippet used to expand the basic sketch have been added here.

config.h

```
...

const char* jquery = "/jquery-3.6.3.min.js";

const char ldrPin = 36;
String sensorValue = "0";

const char ledPin = 2;
String ledValue = "true";
```

In the loop() function, we read the value of the LDR light sensor every second. When using the asynchronous web server, we avoid delays if possible, so we do not use the delay() function. The code fragment below implements a timing with the help of the millis() function, so that the program does not stop running.

Esp32-AsyncWebserver-jQuery-ajax.ino

```
Millis = millis();
if (Millis - previousMillis >= 1000)
{
   previousMillis = Millis;
   sensorValue = String(analogRead(ldrPin));
}
```

amazon

HiLetgo ESP8266 NodeMCU CP2102 ESP-12E Development Board

amazon

ESP8266 Esp-01 Serial Wireless Wifi Transceiver Module Compatible with Arduino

amazon

ESP32 Development Board 2.4 GHz Dual Core WLAN WiFi + Bluetooth

advertising – amazon.com

The following code handles requests sent by the uploaded index.html file.

Esp32-AsyncWebserver-jQuery-ajax.ino

```
/*********   Your server.on code here...   ***********/

  server.on("/", HTTP_GET, [](AsyncWebServerRequest *request)
  {
    request->send(SPIFFS, "/index.html", "text/html", false, processo
  });

  server.on(jquery, HTTP_GET, [](AsyncWebServerRequest *request)
  {
    request->send(SPIFFS, jquery, "text/javascript");
  });

  server.on("/sensorvalue", HTTP_GET, [](AsyncWebServerRequest *reque
  {
    request->send(200, "text/text", sensorValue);
  });

  server.on("/ledvalue", HTTP_GET, [](AsyncWebServerRequest *request)
  {
    request->send(200, "text/text", ledValue);
  });


  server.on("/ledswitch", HTTP_POST, [](AsyncWebServerRequest *reques
  {
    ledValue = request->getParam(0)->value();

    if(ledValue == "true")
    {
      ledValue = "false";
      digitalWrite(ledPin, LOW);
    }
    else if(ledValue == "false")
    {
      ledValue = "true";
      digitalWrite(ledPin, HIGH);
    }
    request->send(200);
  });

  /**************  Your server.on code so far...  *****************
```

The server.on("/", HTTP_GET ....}) function loads the home page, the uploaded index.html file.

The server.on(jquery, HTTP_GET ....}) function serves the jQuery file called in the head section of the index.html file.

index.html

```
<!DOCTYPE html>
<html>
    <head>
        <title>ESP32 asyncWebserver-jQuery-ajax</title>
        <script src="jquery-3.6.3.min.js"></script>
.....
```

The javascript codes are written in the body of the jQuery ready() method, the ready() event occurs when the DOM is loaded.

index.html

```
<script>
  $(document).ready(function() {
    ....
</script>
```

The function server.on("/sensorvalue", HTTP_GET ...}) receives a request every second, and returns the value of the LDR light sensor in response. In the html code, we use jQuery's setInterval() function for this, it runs at intervals specified in milliseconds. We placed the jQuery ajax get() function in the body of the setInterval() function, which retrieves the value of the light sensor from the specified url.

index.html

```
setInterval(function () {
  $.get("/sensorvalue", function(data){
    $('#sensor_result').html(data);
  });
},1000);
```

In the body of the jQuery get() method, the value of the received data is entered into the <div> element with the sensor_result id.

index.html

```
<div id="sensor_result" style="height:40px;"></div>
```

server.on("/ledvalue", HTTP_GET ...}) returns the state of the led, we use this to change the label of the button used to switch the led. The get() function is used to retrieve the state of the led from the server, and then store it in a hidden html element. this value is also passed to a variable. We examine the value of the variable and change the label of the button accordingly.

index.html

```
$.get("ledvalue", function(data) {
  $('#led_status').val(data);
});

var str = $("#led_status").val();
```

```

if(str == "true") {
  $("#ledswitch_button").text("Turn on the led!");}
else {
  $("#ledswitch_button").text("Turn off the led!");
}
```

When the button is pressed in the index.html file, the status of the led is read from the hidden html element in the jQuery click() event handler function, the label of the button is set accordingly, and the new status of the led is sent to the server using the jQuery post() method. After that, we retrieve the current state of the led with the get() function and pass it to the hidden html element.

index.html

```
$("#ledsw itch_button").click(function() {
  str = $("#led_status").val();

  if(str == "true") {
    $("#ledswitch_button").text("Turn on the led!");  }
  else {
    $("#ledswitch_button").text("Turn off the led!");  }
  $.post("ledswitch", {
    led: str
  });
  $.get("ledvalue", function(data) {
    $('#led_status').val(data);
  });
});
```

The function server.on("/ledswitch", HTTP_POST, ….}) receives the event sent by the button used to switch the led. This is where the led is turned on or off, and the value of the variable storing the state of the led is set.

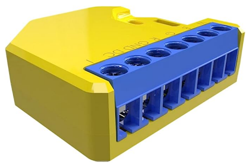
Although jQuery can do much more than what is shown in this example, I hope it helps you understand how jQuery ajax and ESPAsyncWebServer work together.

amazon

Shelly 1 Plus Bluetooth and WiFi Relay

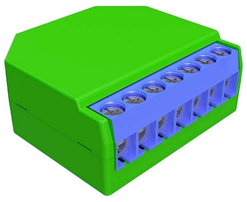
advertising – Amazon.com

amazon



Shelly RGBW 2 Controller, WiFi Smart Remote, LED Light Strip Dimmer

amazon



Shelly Dimmer 2, Relay Switch, WiFi Smart Home Automation, Compatible with Alexa & Google Home

amazon

Shelly 2.5 Double Relay Switch and Roller Shutter WiFi