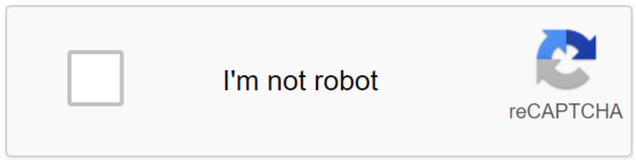

Continue

Esp32 web server with arduino ide pdf

With your script style, can you help me script ESP32 to give feedback when the client connect to our server? Thanks on November 8, 2019 at 8:03am the recently launched successor to ESP8266 - ESP32 was a growing star among IoT or WiFi related projects. It is an extremely cost-effective WiFi module that - with little extra effort - can be programmed to create a standalone web server. How cool it is! A web server is a place that stores, processes and delivers web pages to web customers. A web client is nothing but a web browser on our laptops and smartphones. The communication between the client and the server is made using a special protocol called Hypertext Transfer Protocol (HTTP). In this protocol, the client initiates communication by requesting a specific web page using HTTP and the server responds with the contents of that web page or error message if they are unable to do so (e.g., the known 404 Error). The pages supplied by the server are mostly HTML documents. ESP32 Operating Modes One of esp32's greatest features ensures that it can't only connect to an existing WiFi network and act as a web server, but it can also customize its own network, allowing other devices to connect directly to it and access web pages. This is possible because the ESP32 can run in three different modes: station mode, Soft Access Point mode, and both at the same time. This makes it possible to build grid networks. The station (STA) ModeThe ESP32, which connects to an existing WiFi network (one created wireless router) is called the station (STA)In STA ESP32 mode receives IP from the wireless router to which it is connected. With this IP address, it can set up a web server and deliver web pages to all connected devices within the existing WiFi network. Soft Access Point (AP) ModeThe ESP32, which creates its own WiFi network and acts as a hub (just like a WiFi router) for one or more stations called the access point (AP). Unlike the WiFi router, it does not have an interface for the wired network. Thus, this mode of operation is called Soft Access Point (soft-AP). Also, the maximum number of stations that can connect to it is limited to five. In AP mode, ESP32 creates a new WiFi network and installs SSID (network name) and IP address to it. With this IP address, it can deliver web pages to all connected devices within its own network. Wiring - Connecting LEDs to ESP32Now that we know the basics of how a web server works, and in what mode ESP32 can create a web server, it's time to connect some LEDs to ESP32 that we want to control WiFi.Start by placing ESP32 on your board, that each side of the board is on a separate side of the board. Then connect the two LEDs to the digital GPIO 4 and 5 via the 220th current, limiting the resistor. When you are done you should have what looks like the illustration shown below. Wiring LEDs to ESP32Concept behind behind Things from ESP32 Web ServerSo, you may think: How will I control things with a web server that just processes and delivers web pages? Well, then you need to understand what's going on behind the scenes. When you type the URL into a web browser and tap ENTER, the browser sends the HTTP request (as well as the GET request) to the web server. It's the web server's job to process this query by doing something. You may have already realized that we will control things, access to a certain URL. For example, let's say we typed URL as a browser. The browser then sends the HTTP request to ESP32 to process the request. When ESP32 reads this request, it knows that the user wants to turn on the ON LED. Thus, it transforms the ON LED and sends a dynamic web page to the browser showing THE status: ON. As simple as a pie! ESP32 as HTTP Server via WiFi Access Point (AP) modeNow let's move on to interesting things! As the headline suggests, this example demonstrates how to turn ESP32 into an access point (AP) and serve web pages to any connected customer. First, connect ESP32 to your computer and try the sketches; and then we'll dissect it in some detail.#include #include / Put your SSID and password qlt.h'gt; qlt.h'gt; const char ssid ESP32; Enter SSID here const charpassword 12345678; Enter the password here /Put the IP Address Details (/ IPAddress local_ip (192,168,1); IPAddress Gateway (192,168,1,1); IPAddress (255,255,255.0); WebServer server (80); uint8_t LED1pin No 4; bool LED1status - LOW; uint8_t LED2pin No 5; bool LED2status - LOW; Void Setting - 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 (НАЧАЛ работу сервера HTTP); - void cycle () - server.handleClient (); If (LED1status) digitalWrite (LED1pin, HIGH handle_OnConnect); LED2status - LOW; Serial.println (GPIO4 Status: OFF STATUS GPIO5: OFF); server.send (200, text/html, SendHTML (LED1status,LED2status)); - invalid handle_led1on () - LED1status - HIGH; Serial.println (GPIO4 Status: ON); server.send (200, text/html, SendHTML (admittedly, LED2status)); - invalid handle_led1off -- LED1status - LOW; Serial.println (GPIO4 Status: OFF); server.send (200, text/html, SendHTML (false,LED2status)); - invalid handle_led2on () - LED2status - HIGH; Serial.println (Статус GPIO5: ON); server.send (200, text/html, SendHTML (LED1status, true)); Void- LED2status - LOW; Serial.println (Статус GPIO5: OFF); server.send (200, текст/html, SendHTML(LED1status, false)); недействительный handle_NotFound ()) server.send (404, текст/простой, Не найдено); Строка SendHTML (uint8_t led1stat,uint8_t led2stat) <! DOCTYPE html><html>; ptr q<head><meta name=\viewport\ content=\width=device-width, initial-scale=1.0, user-scalable=no\>; ptr q<title>Светодиодный контроль</title>; ptr <style>html { font-family: Helvetica; display: inline-block; margin: 0px auto; text-align: center;}; ptr +=body{margin-top: 50px;} h1 {color: #444444;margin: 50px auto 30px;} h3 {color: #444444;margin-bottom: 50px;}; ptr +=.button {display: block;width: 80px;background-color: #3498db;border: none;color: white;padding: 13px 30px;text-decoration: none;font-size: 25px;margin: 0px auto 35px;cursor: pointer;border-radius: 4px;}; ptr +=.button-on {background-color: #3498db;}; ptr +=.button-on:active {background-color: #2980b9;}; ptr +=.button-off {background-color: #34495e;}; ptr +=.button-off:active {background-color: #2c3e50;}; ptr +=p {font-size: 14px;color: #888;margin-bottom: 10px;}; ptr +=</style> q ; ptr </head>; ptr q<body>; ptr q<h1>Веб-сервер ESP32</h1>; ptr q<h3>Использование режима точки доступа (AP)</h3>; если (led1stat) «ptr»<p>Статус LED1: ON</p>HET; еще «ptr»<p>LED1 Статус: ВЫКЛЮЧЕН</p>НА; если (led2stat) «ptr»<p>Статус LED2: ON</p>HET; еще «ptr»<p>LED2 Статус: ВЫКЛЮЧЕН</p>HA; ptr q</body>; ptr q</html>; возврат ptr; «Доступ к веб-серверу в режиме APПоседи загрузки эскиза откройте серийный монитор со скоростью 115200. И нажмите кнопку RESET на ESP32. Если все в порядке, он покажет http сервер начал сообщение. Далее, найти любое устройство, которое вы можете подключить к сети WiFi - телефон, ноутбук и т.д. И ищите сеть под названием ESP32. Присоединяйтесь к сети с паролем 123456789.После подключения к вашей сети ESP32 AP, загрузите браузер и указать его на 192.168.1.1 ESP32 должны служить до веб-страницы с указанием текущего состояния светодиодов и две кнопки, чтобы контролировать их. Если одновременно взглянуть на серийный монитор, то можно увидеть статус контактов GPIO ESP32. Теперь нажмите кнопку, чтобы включить LED1 ON, следя за URL. Как только вы нажмете на кнопку, ESP32 получает запрос на URL /led1on. Затем он поворачивает LED1 ON и служит веб-страницы со статусом светодиода обновляется. Он также печатает статус штифта GPIO на серийном мониторе. Вы можете протестировать кнопку LED2 и проверить, что она работает аналогичным образом. Теперь Take a closer look at the code to see how it works so you can change it to meet your needs. A detailed explanation of the CodeStitch begins with the inclusion of the inclusion Library. This library provides ESP32 with specific WiFi methods we call to connect to the network. After that, we also include a webServer.h library that has some available methods that will help us build a server and handle incoming http requests without worrying about the low level of implementation details.#include of the wifi.h'gt;#include How to install ESP32 in Access Point (AP) mode, it will create a WiFi network. So we have to set it SSID, password, IP address, IP subnet mask and IP gateway./ Enter SSID here const char' password No 12345678; Enter the password here /- Please click on IPAddress local_ip (192,168,1,1); IPAddress Gateway (192,168,1,1); IPAddress (255,255,255.0); Next, we announce the webServer library facility so we can access its features. The designer of this object takes the port (where the server will listen) as a parameter. Since 80 is the default port for HTTP, we will use this value. You can now access the server without having to specify the port in the URL./ Declare the webServer object of the WebServer server library (80); Next, we announce the ESP32 in GPIO pins to which LEDs are connected and their initial state.uint8_t LED1pin No 4; bool LED1status - LOW; uint8_t LED2pin No 5; bool LED2status - LOW; We set up our HTTP server before actually launching. First of all, we open a serial connection for debugging and install GPIO ports for OUTPUT. Serial.beginning (115200); pinMode (LED1pin, OUTPUT); pinMode (LED2pin, OUTPUT); We then created a soft access point to create a Wi-Fi network, proving SSID, password, IP address, IP network mask, and IP gateway. WiFi.softAP (ssid, password); WiFi.softAPConfig (local_ip, gateway, subnet); Delay (100); To process incoming HTTP requests, you need to specify what code to run when hitting a particular URL. That's why we use the method. This method takes two parameters. The first is the URL path, and the second is the name of the function that we want to perform when that URL is hit. For example, the first line below the code fragment indicates that when a server receives a HTTP request on a root (// trajectory), it triggers the handle_OnConnect function. Note that this URL is a relative way. In addition, we must provide 4 more URLs to handle two 2 LEDs.server.on states (/, handle_OnConnect); server.on (/led1on, handle_led1on); server.on (/led1off, handle_led1off); server.on (/led2on, handle_led2on); server.on (/led2off, handle_led2off); We did not specify what the server should do if the client requested any URL other than the server.on() specified. It must meet the status of HTTP 404 (Not found) and a message to the user. We put this into function as well, and use server.onNotFound () to tell him that he should ero, когда он получает запрос на</WebServer.h> </WiFi.h> </WiFi.h> this has not been stated with server.onserver.onNotFound (handle_NotFound); Now, to start our server, we call the start method on the server object. Serial.println (НАЧАЛ работу сервера HTTP); To process actual incoming HTTP requests, we need to call the handleClient method on the server object. We also change the state of the LED according to the request.void cycle () - server.handleClient(); If (LED1status) digitalWrite (LED1pin, HIGH); else digitalWrite(LED1pin, LOW); If (LED2status) digitalWrite (LED2pin, HIGH); Next, we need to create a function that we attached to the root (/) URL with server.on. Remember? At the beginning of this function, we state both LEDs on LOW (the initial state of LEDs) and print it on a serial monitor. We use the sending method to respond to the HTTP query. Although the method can be called a different set of arguments, its simplest form consists of the HTTP response code, the type of content and content. In our case, we send code 200 (one of the HTTP status codes) that corresponds to the OK answer. We then specify the type of content as text/html, and finally we call SendHTML a user function that creates a dynamic HTML page containing LEDs.void status handle_OnConnect () LED2status - LOW; Serial.println (GPIO4 Status: OFF STATUS GPIO5: OFF); server.send (200, text/html, SendHTML (LED1status,LED2status)); In addition, we need to create four features to handle LED ON/OFF queries and 404 page.void handle_led1on () Serial.println (GPIO4: ON); server.send (200, text/html, SendHTML (admittedly, LED2status)); - invalid handle_led1off -- LED1status - LOW; Serial.println (GPIO4 Status: OFF); server.send (200, text/html, SendHTML (false,LED2status)); - invalid handle_led2on () - LED2status - HIGH; Serial.println (Статус GPIO5: ON); server.send (200, text/html, SendHTML (LED1status, true)); invalid handle_led2off - LED2status - LOW; Serial.println (GPIO5 Status: OFF); server.send (200, text/html, SendHTML (LED1status, false)); invalid handle_NotFound.) server.send (404, text/simple, not found); The HTML Web PageSendHTML () feature is responsible for creating a web page whenever the ESP32 web server receives a request from a web client. It just concatenates HTML code into a big line and goes back to the server.send () features we discussed earlier. The feature accepts LED status as a setting for dynamic HTML content creation. The first text that you should always zlt;! DocTYPE'gt; send, it is a declaration that indicates that we are sending HTML code. The SendHTML line (uint8_t led1stat,uint8_t led2stat) DOCTYPE html><html>; Next, the viewport makes the web page responsive in any web browser. While the title tag sets page.ptr<head><meta ></meta ></head></html> ></head></html> The contents of the width of the device, the initial scale -1.0, the user-scalable no; ptr q'lt;title Stacking Web PageNext, we have CSS to style the buttons and the look of the web page. We select the Helvetica font, identify the content that will appear as a row block and align in the center.задержка (100); pinMode (LED1pin, OUTPUT); pinMode (LED2pin, OUTPUT); Serial.println (Подключение к); Serial.println (ssid); подключение к локальной сети Wi-Fi WiFi.begin (ssid, пароль); проверить Wi-Fi подключен к сети Wi-Fi в то время как (WiFi.status() ! WL_CONNECTED) задержка (1000); Serial.print (.); - Serial.println (); Serial.println (WiFi подключен..!); Serial.print (Got IP:); Serial.println(WiFi.localIP();; 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 (НАЧАЛ работу сервера HTTP); - пустотный цикл () - server.handleClient (); если (LED1status) digitalWrite (LED1pin, HIGH handle_OnConnect); LED2статус - LOW; Serial.println (Статус GPIO4: OFF СТАТУС GPIO5: OFF); server.send (200, текст/html, SendHTML (LED1status,LED2status)); - недействительные handle_led1on () - LED1status - HIGH; Serial.println (Статус GPIO4: ON); server.send (200, текст/html, SendHTML (правда,LED2status)); - недействительные handle_led1off () - LED1status - LOW; Serial.println (Статус GPIO4: OFF); server.send (200, текст/html, SendHTML (ложный,LED2status)); - недействительные handle_led2on () - LED2status - HIGH; Serial.println (Статус GPIO5: ON); server.send (200, текст/html, SendHTML(LED1status, true)); недействительные handle_led2off () - LED2status - LOW; Serial.println (Статус GPIO5: OFF); server.send (200, текст/html, SendHTML(LED1status, false)); недействительный handle_NotFound ()) server.send (404, текст/простой, Не найдено); Строка SendHTML (uint8_t led1stat,uint8_t led2stat) <! DOCTYPE html><html>; ptr q<head><meta name=\viewport\ content=\width=device-width, initial-scale=1.0, user-scalable=no\>; ptr q<title>Светодиодный контроль</title>; ptr <style>html { font-family: Helvetica; display: inline-block; margin: 0px auto; text-align: center;}; ptr +=body{margin-top: 50px;} h1 {color: #444444;margin: 50px auto 30px;} h3 {color: #444444;margin-bottom: 50px;}; ptr +=.button {display: block;width: 80px;background-color: #3498db;border: none;color: white;padding: 13px 30px;text-decoration: none;font-size: 25px;margin: 0px auto 35px;cursor: pointer;border-radius: 4px;}; ptr +=.button-on {background-color: #3498db;}; ptr +=.button-on:active {background-color: #2980b9;}; ptr +=.button-off {background-color: #34495e;}; ptr +=.button-off:active {background-color: #2c3e50;}; ptr +=p {font-size: 14px;color: #888;margin-bottom: 10px;}; ptr +=</style> q ; ptr </head>; ptr q<body>; ptr q<h1>Веб-сервер ESP32</h1>; ptr q<h3>Режим Станция (STA)</h3>; если (led1stat) «ptr»<p>Статус LED1: ON</p>HET; Еще Status: WINNER If (led2stat) ptr Still ptr ptr z; ptr z; Return ptr; Access to the web server in STAPos mode after downloading the sketch, open a serial monitor at a speed of 115,200. And click the RESET button on ESP32. If all is well, it will output a dynamic IP address received from your router and show the HTTP server started the message. Then download the browser and bring it to the IP address shown on the serial monitor. ESP32 must maintain a web page showing the current state of LEDs and two buttons to control them. If you look at the serial monitor at the same time, you can see the status of GPIO ESP32 contacts. Now click to turn on LED1 ON, keeping an eye on the URL. As soon as you press the button, ESP32 receives a request for URL/led1on. It then rotates the LED1 ON and serves a web page with LED status updated. It also prints the status of the GPIO pin on a serial monitor. You can test the LED2 button and check that it works in a similar way. Explaining the codeIf you follow this code with the previous code, the only thing that we do not install is a soft access point, instead we join the existing network using wiFi.begin.//connect to your local Wi-Fi network WiFi.start (ssid, password); While ESP32 tries to connect to the network, we can check the state of the connection with the WiFi.status feature./)the feature./check wi-fi is connected to the Wi-Fi network, while (WiFi.status)) WL_CONNECTED ! Serial.print (.); Just for your information, this feature returns the following statuses: WL_CONNECTED: assigned when you connect to Wi-Fi networkWL_NO_SHIELD: assigned when there is no Wi-Fi shield presentWL_IDLE_STATUS: temporary status assigned when WiFi.begin() is called and remains active until the number of attempts expires (as a result of WL_CONNECT_FAILED) or connection is established (as a result of WL_CONNECTED) WL_NO_SSID_AVAIL: when there is no SSID availableWL_SCAN_COMPLETED: appointed when the scanning network completedWL_CONNECT_FAILED: appointed when the connection fails for all attemptsWL_CONNECTION_LOST: assigned when the connection lostWL_DISCONNECTED: assigned when disconnected from the networkE 200, ESP32 is connected to the network, the sketch prints the IP address assigned to ESP32, displaying the value of WiFi.localIP () on a serial monitor. Serial.println (); Serial.println (WiFi connected..!); Serial.print (Got IP:); Serial.println (WiFi.localIP();; the only difference between AP mode and STA is that one creates a network and the other joins the existing network. part of the code to handle HTTP requests and maintain a web page in STA mode is the same as that of Mode explained above. This includes: Ad GPIO contacts ESP32, to which LEDs are connectedIn addition server.on () methods of processing incoming queries httpDefining server.onNotFound () method for processing HTTP 404 errorCreation of custom functions that are performed when a specific URL hitCreating HTML pageStyling webpagesCretyling the creation and buttons displaying their status esp32 web server with arduino ide pdf. esp32 web server with arduino ide download. esp32 servo motor web server with arduino ide. esp32 with dht11 dht22 temperature humidity web server using arduino ide

d496e1bea36d8c2.pdf