How to set a Static or Fixed IP Address on your ESP32 Tutorials Basic Electronics Arduino ESP32 ESP8266 ESP32 Basics ESP32 Getting Started ESP32 Arduino IDE Setup ESP32 Pinout Reference WROOM-32 Pinout Reference ESP32 ADC ESP32 Hall Sensor ESP32 Touch Pins ESP32 PWM ESP32 GPIO Interrupts ESP32 Sleep Modes ESP32 Deep Sleep Protocols ESP32 HTTP Web Server ESP32 WebSocket Server ESP32 Bluetooth Classic ESP32 Bluetooth LE Web Servers Web Server Getting Started DHT11/DHT22 Web Server BME280 Web Server DS18B20 Web Server ESP32-CAM ESP32-CAM Getting Started ESP32-CAM Pinout Reference ESP32 Projects ESP32 WLED ESP32 WLED Sound-Reactive Useful Guides ESP32 MAC Address ESP32 Fatal Error Fix ESP32 mDNS ESP32 Static IP Address ESP32 OTA Updates ESP32 OTA Web Updater ESP32 NTP Server Displays ESP32 with OLED Display ESP32 with I2C LCD Sensors ESP32 DHT11/DHT22 ESP32 BME280 ESP32 Multiple DS18B20 Table Of Contents Introduction Step 1 - Set Up the Arduino IDE Step 2 - Connect the ESP32 to your Computer Step 3 - Obtain Current Network Settings Step 4 - Set a Static IP Address Step 6 - Testing Detailed Code Explanation ESP32 How to set a Static or Fixed IP Address on your ESP32 If you’ve ever tried running a web server on an ESP32, you’ve likely noticed something annoying—every time you restart your ESP32, the IP address can change, based on what the router decides to assign at the moment. This means you always have to check the serial monitor to find out this new IP address. It’s quite a hassle, right? This is where static IP addresses come into play. By setting a static IP address, you can access the web server using the same IP address, even after restarting the ESP32. This is also useful to avoid confusion when you have multiple ESP32s connected to your network. Fortunately, once you know how, it’s a fairly simple and quick process to set a static IP address. We’ll guide you step-by-step on how to set a static IP address on your ESP32. Let’s dive in! Step 1 – Set Up the Arduino IDE We will be using the Arduino IDE to program the ESP32, so please ensure you have the ESP32 add-on installed before you proceed: Installing ESP32 Board in the Arduino IDE There are several development platforms available for programming the ESP32. You can go with: Arduino IDE – intended for those who are familiar with ArduinoEspruino... Step 2 – Connect the ESP32 to your Computer Use a micro-USB cable to connect your ESP32 board to your computer. Step 3 – Obtain Current Network Settings Before setting a static IP, it’s a good practice to check the current network settings (IP, Gateway, Subnet, and DNS) assigned by your router. This information can help avoid IP conflicts. To find out the ESP32’s current IP address and other network settings, you need to upload the following sketch to your ESP32. Before that, there is one essential modification you need to make. Update the following two variables with your actual WiFi SSID and password. const char \* ssid = "YourNetworkName" ; // Enter SSID here const char \* password = "YourPassword" ; //Enter Password here After making these changes, go ahead and upload the code. #include <WiFi.h> // Replace with your network credentials const char \* ssid = "YourNetworkName" ; // Enter SSID here const char \* password = "YourPassword" ; //Enter Password here void setup () {  
 Serial. begin ( 115200 );  
   
 Serial. print ( "Connecting to " );  
 Serial. println (ssid); // Connect to Wi-Fi WiFi. begin (ssid, password); while (WiFi. status () != WL\_CONNECTED) { delay ( 1000 );  
 Serial. print ( "." );  
 }  
   
 Serial. println ( "" );  
 Serial. println ( "Connected..!" );  
   
 Serial. print ( "Current ESP32 IP: " );  
 Serial. println (WiFi. localIP ());  
 Serial. print ( "Gateway (router) IP: " );  
 Serial. println (WiFi. gatewayIP ());  
 Serial. print ( "Subnet Mask: " );  
 Serial. println (WiFi. subnetMask ());  
 Serial. print ( "Primary DNS: " );  
 Serial. println (WiFi. dnsIP ( 0 ));  
 Serial. print ( "Secondary DNS: " );  
 Serial. println (WiFi. dnsIP ( 1 ));  
} void loop () {  
} After uploading the code, open the Serial Monitor and set the baud rate to 115200. Then, press the EN button on the ESP32. It may take a few moments to connect to your network, after which it will print the current network settings of the ESP32 to the serial monitor. Take note of these. Step 4 – Set a Static IP Address After obtaining the current network settings, you can now set a static IP address. Modify the sketch below to include the static IP configuration. You will need to specify the static IP address, Gateway, Subnet Mask, and DNS settings (optional). IPAddress staticIP ( 192 , 168 , 1 , 100 ); // ESP32 static IP IPAddress gateway ( 192 , 168 , 1 , 1 ); // IP Address of your network gateway (router) IPAddress subnet ( 255 , 255 , 255 , 0 ); // Subnet mask IPAddress primaryDNS ( 192 , 168 , 1 , 1 ); // Primary DNS (optional) IPAddress secondaryDNS ( 0 , 0 , 0 , 0 ); // Secondary DNS (optional) Based on the current network settings, select an appropriate static IP address for your ESP32. This address should be within the same subnet as your current network but outside the range your router typically assigns (to avoid IP conflicts). Once you modify the sketch, upload it to the ESP32. #include <WiFi.h> // Replace with your network credentials const char \* ssid = "Your\_SSID" ; const char \* password = "Your\_PASSWORD" ; // Static IP configuration IPAddress staticIP ( 192 , 168 , 1 , 100 ); // ESP32 static IP IPAddress gateway ( 192 , 168 , 1 , 1 ); // IP Address of your network gateway (router) IPAddress subnet ( 255 , 255 , 255 , 0 ); // Subnet mask IPAddress primaryDNS ( 192 , 168 , 1 , 1 ); // Primary DNS (optional) IPAddress secondaryDNS ( 0 , 0 , 0 , 0 ); // Secondary DNS (optional) void setup () {  
 Serial. begin ( 115200 ); // Connect to Wi-Fi WiFi. begin (ssid, password); while (WiFi. status () != WL\_CONNECTED) { delay ( 1000 );  
 Serial. println ( "Connecting to WiFi..." );  
 } // Configuring static IP if (!WiFi. config (staticIP, gateway, subnet, primaryDNS, secondaryDNS)) {  
 Serial. println ( "Failed to configure Static IP" );  
 } else {  
 Serial. println ( "Static IP configured!" );  
 }  
   
 Serial. print ( "ESP32 IP Address: " );  
 Serial. println (WiFi. localIP ()); // Print the ESP32 IP address to Serial Monitor } void loop () { // Nothing to do here } Step 6 – Testing After uploading the code to your ESP32, use the Serial Monitor to confirm that the ESP32 is now using the static IP address you set. Static IP vs mDNS While setting a static IP for your ESP32 is effective, there’s another approach worth considering: mDNS (Multicast DNS). This method offers significant advantages over static IP. mDNS allows you to access the web server running on your ESP32 using a user-friendly hostname, such as “esp32.local”, rather than fiddling with an IP address. Moreover, even if the IP address of your ESP32 changes, mDNS automatically resolves the new IP address to the same hostname. This means you can keep using the same hostname without worrying about tracking IP address changes. To learn more about how to implement mDNS on your ESP32, check out our detailed guide here:. How to set up mDNS on an ESP32 If you have a single ESP32 on your network, you can connect to it using its IP address (192.168.1.128, for example). But imagine having several... Detailed Code Explanation The sketch begins by including the WiFi.h library. This library provides the functions needed to connect the ESP32 to a Wi-Fi network and set its IP configuration. #include <WiFi.h> Next, constants for the Wi-Fi SSID ( ssid ) and password ( password ) are defined. You’ll need to replace these with your actual Wi-Fi network’s name and password. const char \* ssid = "Your\_SSID" ; const char \* password = "Your\_PASSWORD" ; Then, IPAddress objects are created for the static IP, gateway, subnet mask, and optionally, primary and secondary DNS servers. These objects will store the respective addresses and are used later to configure the ESP32’s network settings. // Static IP configuration IPAddress staticIP ( 192 , 168 , 1 , 100 ); // ESP32 static IP IPAddress gateway ( 192 , 168 , 1 , 1 ); // IP Address of your network gateway (router) IPAddress subnet ( 255 , 255 , 255 , 0 ); // Subnet mask IPAddress primaryDNS ( 192 , 168 , 1 , 1 ); // Primary DNS (optional) IPAddress secondaryDNS ( 0 , 0 , 0 , 0 ); // Secondary DNS (optional) Inside the setup() function, serial communication is initiated with the computer. Serial. begin ( 115200 ); The ESP32 attempts to connect to the Wi-Fi network using WiFi.begin(ssid, password) . A while loop ensures the ESP32 keeps trying to connect until it succeeds. WiFi. begin (ssid, password); while (WiFi. status () != WL\_CONNECTED) { delay ( 1000 );  
 Serial. print ( "." );  
} Once connected to Wi-Fi, the ESP32 is configured to use the previously defined static IP, gateway, subnet mask, and DNS servers using the WiFi.config() function. If there’s an issue configuring the static IP, the message “Failed to configure Static IP” is printed to the Serial Monitor. if (!WiFi. config (staticIP, gateway, subnet, primaryDNS, secondaryDNS)) {  
 Serial. println ( "Failed to configure Static IP" );  
} else {  
 Serial. println ( "Static IP configured!" );  
} After setting the IP configuration, the ESP32’s current IP address is printed to the Serial Monitor using WiFi.localIP() . If everything goes as planned, this should display the static IP address that was set. Serial. print ( "ESP32 IP Address: " );  
Serial. println (WiFi. localIP ()); The IP is set in the setup() function, and since there’s nothing left to do, the loop() function is kept empty. void loop () {  
} SHARE Disclaimer Privacy Policy About Contact Us Copyright © 2023 LastMinuteEngineers.com. All rights reserved. x x



































