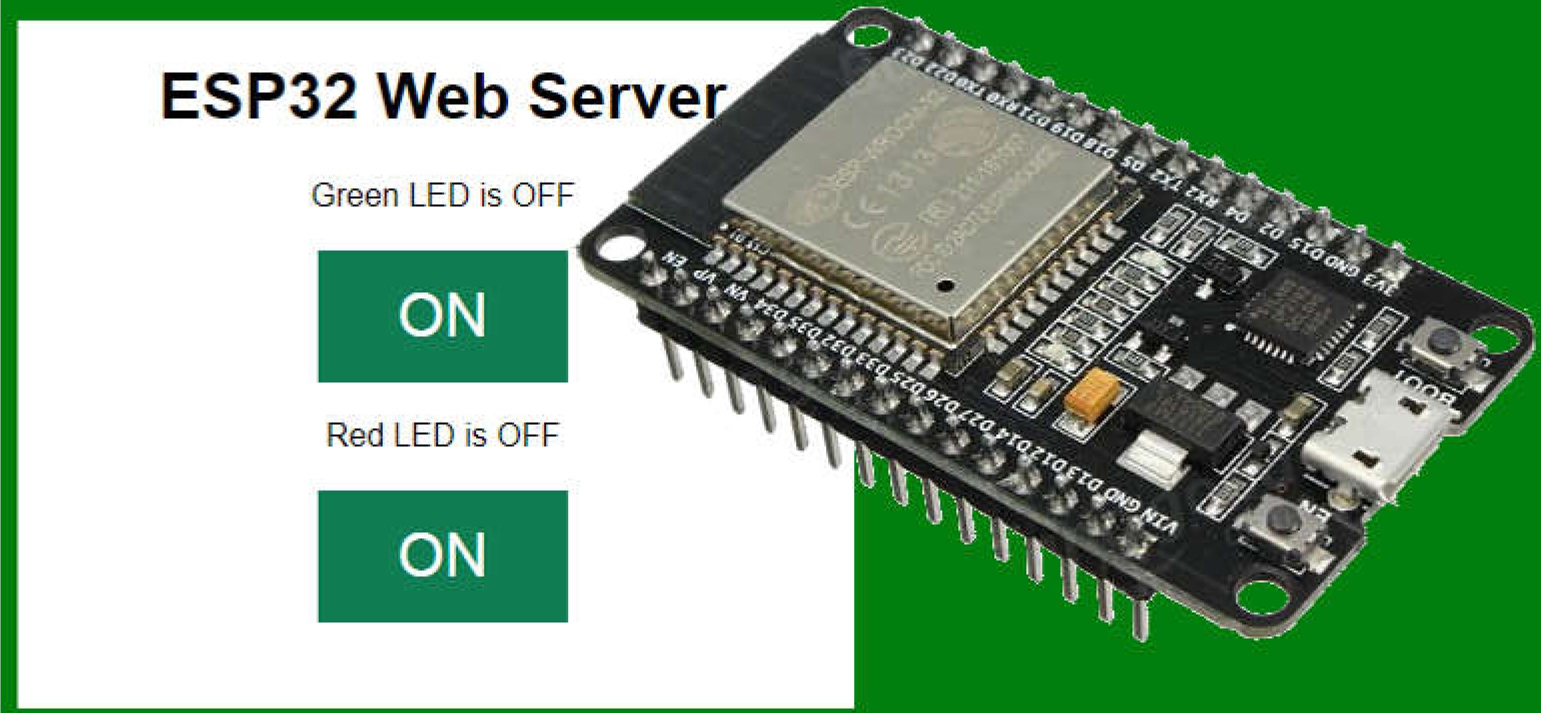
ESP32 Web Server



ESP32 Web Server

Green LED is OFF

ON

Red LED is OFF

ON

D23 D22 TX0 RX0 D21 D19 D18 D5 TX2 RX2 D4 D2 D15 GND 3V3
EN VP VN D34 D35 D32 D33 D25 D26 D27 D14 D12 D13 GND VIN
BOOT
EN

Step By Step

by Alexander Chukhryaev

1st edition

INDEX

Disclaimer

This eBook has been written for information purposes only. Every effort has been made to make this eBook as complete and accurate as possible.

The purpose of this eBook is to educate. It's a first edition of this book.

The author (Alexander Chukhryaev) does not warrant that the information contained in this eBook is fully complete and shall not be responsible for any errors or omissions.

The author (Alexander Chukhryaev) shall have neither liability nor responsibility to any person or entity with respect to any loss or damage caused or alleged to be caused directly or indirectly by this eBook.

This eBook contains code examples which you can use on your own projects, excepted where otherwise noted.

You cannot redistribute this eBook.

This eBook is only available for free download at:
http://acoptex.com/wp/download

Please send an email to the author (Alexander Chukhryaev - info@acoptex.com), if you find this eBook anywhere else.

Introduction

This eBook will help you to build a web server with the ESP32 WiFi and Bluetooth development board.

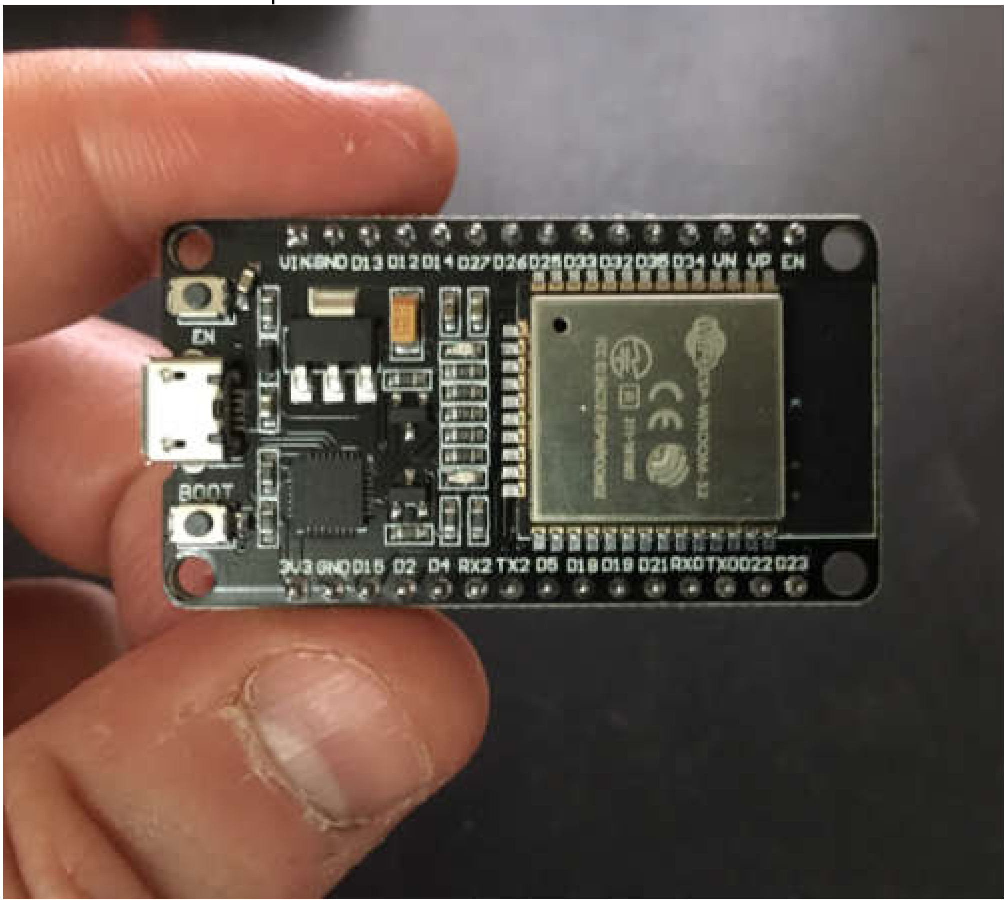


Image credit : Acoptex

I will use the ESP32 DEVKIT V1 DOIT board, but any other ESP32 development board with WiFi, Bluetooth and the ESP-WROOM-32 chip should work well too.

Whether you are building a robot or working with Arduino, knowing how to use ESP32 development board will come in handy.

Have fun with your projects. Thank you for reading,
P.S. Make sure you visit our website to see the latest projects!
http://acoptex.com

Connect with Acoptex.Com

If you have any questions, please don't hesitate to contact us.

Here are some ways to stay in touch.

Visit our website
(http://acoptex.com)

Subscribe on YouTube
(https://www.youtube.com/channel/UCPQwh8eONWeNUeZ24CqDR1g/)

Like on facebook
(https://www.facebook.com/acoptexco/)

Follow on Twitter
(https://twitter.com/Acoptexcom/)

Check us on GitHub
(https://github.com/AcoptexCom/)

Follow us on Instagram
(https://www.instagram.com/acoptexcom/)

Understanding The ESP32 development board

ESP32 is a series of low cost, low power system on a chip microcontrollers with integrated Wi-Fi and dual-mode Bluetooth. The ESP32 series employs a Tensilica Xtensa LX6 microprocessor in both dual-core and single-core variations and includes in-built antenna switches, RF balun, power amplifier, low-noise receive amplifier, filters, and power management modules. ESP32 is created and developed by Espressif Systems, a Shanghai-based Chinese company, and is manufactured by TSMC using their 40 nm process. It is a successor to the ESP8266 microcontroller.

ESP32 can perform as a complete standalone system or as a slave device to a host MCU, reducing communication stack overhead on the main application processor. ESP32 can interface with other systems to provide Wi-Fi and Bluetooth functionality through its SPI / SDIO or I2C / UART interfaces.

ESP32 is highly-integrated with in-built antenna switches, RF balun, power amplifier, low-noise receive amplifier, filters, and power management modules. ESP32 adds priceless functionality and versatility to your applications with minimal Printed Circuit Board (PCB) requirements.

ESP32 is capable of functioning reliably in industrial environments, with an operating temperature ranging from –40°C to +125°C. Powered by advanced calibration circuitries, ESP32 can dynamically remove external circuit imperfections and adapt to changes in external conditions.

Engineered for mobile devices, wearable electronics and IoT applications, ESP32 achieves ultra-low power consumption with a combination of several types of proprietary software. ESP32 also includes state-of-the-art features, such as fine-grained clock gating, various power modes and dynamic power scaling.

You can find this ESP32 development board here: http://s.click.aliexpress.com/e/57cwzU7W

There are a lot of ESP32 development boards:

ESP32 DOIT DEVKIT V1

ESP32-DevKit

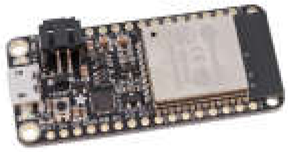
ESP-32S NodeMCU

ESP32 Thing Plus SparkFun

WEMOS LOLIN32

ESP32 WEMOS OLED

ADAFRUIT HUZZAH32



Features of the ESP32 include the following:

Processors:

- CPU: Xtensa dual-core (or single-core) 32-bit LX6 microprocessor, operating at 160 or 240 MHz and performing at up to 600 DMIPS
- Ultra low power (ULP) co-processor
- Memory: 520 KiB SRAM

Wireless connectivity:

- Wi-Fi: 802.11 b/g/n
- Bluetooth: v4.2 BR/EDR and BLE

Peripheral interfaces:

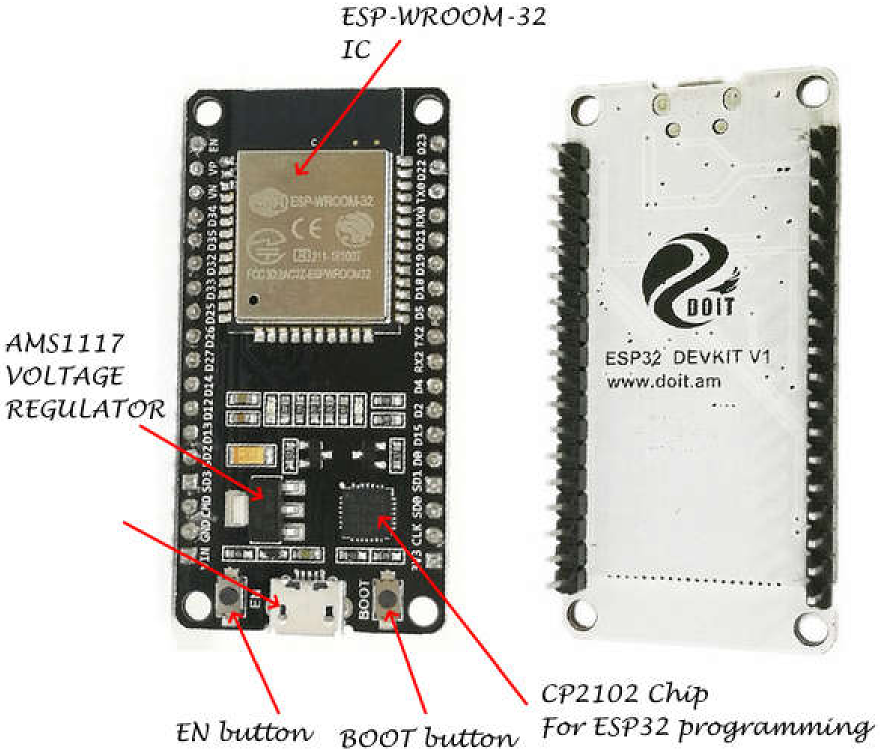
- 12-bit SAR ADC up to 18 channels
- 2 × 8-bit DACs
- 10 × touch sensors (capacitive sensing GPIOs)
- Temperature sensor
- 4 × SPI
- 2 × I²S interfaces
- 2 × I²C interfaces
- 3 × UART
- SD/SDIO/CE-ATA/MMC/eMMC host controller
- SDIO/SPI slave controller
- Ethernet MAC interface with dedicated DMA and IEEE 1588 Precision Time Protocol support
- CAN bus 2.0
- Infrared remote controller (TX/RX, up to 8 channels)
- Motor PWM
- LED PWM (up to 16 channels)

- Hall effect sensor
- Ultra low power analog pre-amplifier

Power management:

- Internal low-dropout regulator
- Individual power domain for RTC
- 5uA deep sleep current
- Wake up from GPIO interrupt, timer, ADC measurements, capacitive touch sensor interrupt

We will be using the ESP32 DEVKIT V1 DOIT board in this EBook.

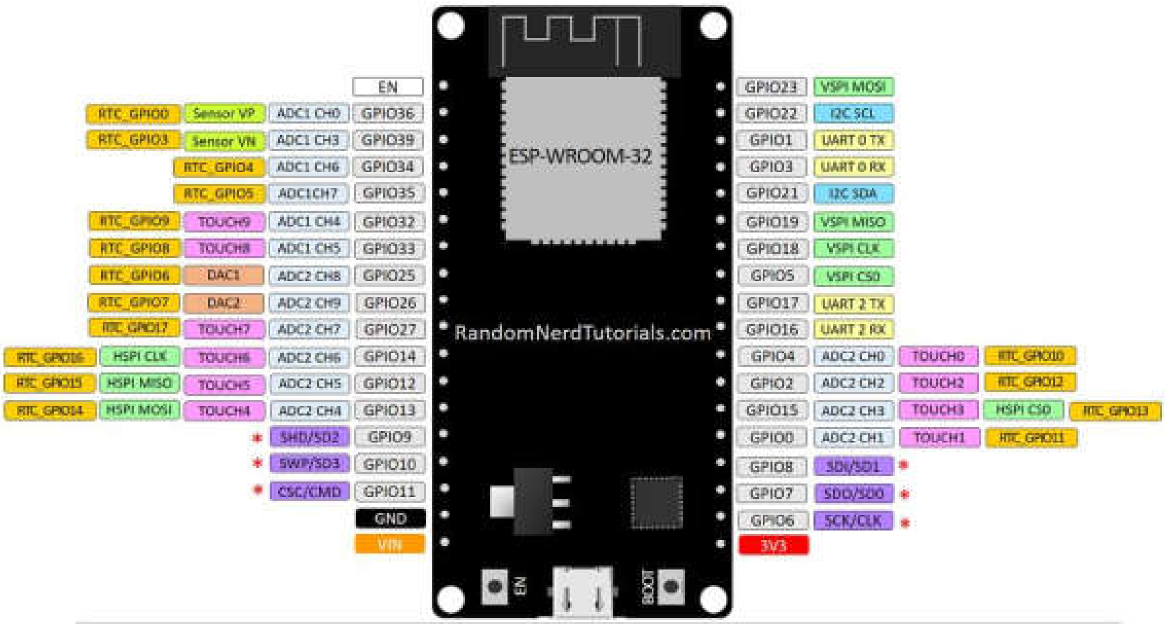


So, what can you do with this module? You can:

- create a web server
- send HTTP requests
- control outputs
- read inputs and interrupts
- build IoT gadgets
- and so on.

ESP32 DEVKIT V1 DOIT Board Pinout

version with 36 GPIOs



* Pins SCK/CLK, SDO/SD0, SDI/SD1, SHD/SD2, SWP/SD3 and SCS/CMD, namely, GPIO6 to GPIO11 are connected to the integrated SPI flash integrated on ESP-WROOM-32 and are not recommended for other uses.

Image credit: RandomNerdTutorials.com

version with 30 GPIOs

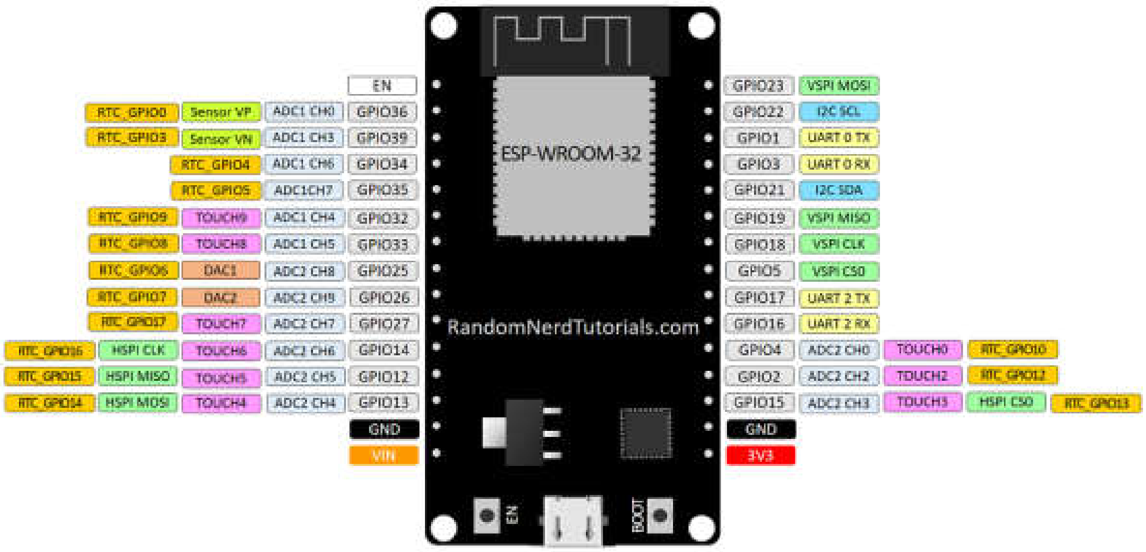


Image credit: RandomNerdTutorials.com

Arduino IDE

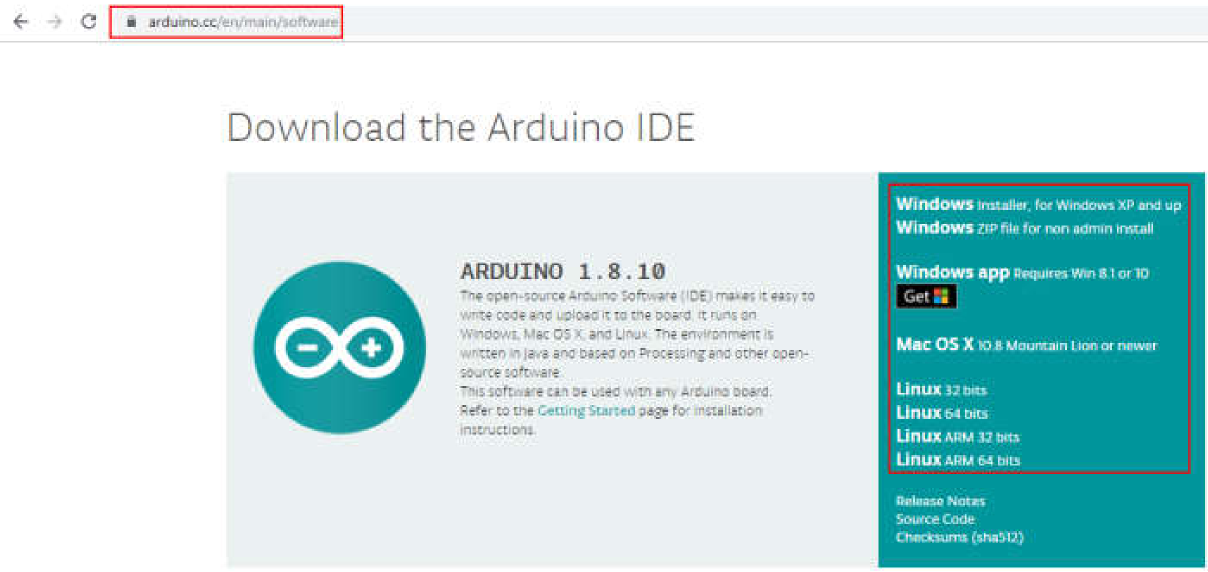
The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software. This software can be used with any Arduino board, ESP8266 WiFi module, ESP32 WiFi module. The Arduino IDE is a multiplatform software, which means that it runs on Windows, Mac OS X or Linux (it was created in JAVA). First you need to download, install and prepare your Arduino IDE to work with the ESP8266 WiFi module. Then you can program your ESP8266 WiFi module using the simple C programming language.

Preparations:

You need to have the JAVA installed in your computer (PC). If you do not have it, go to this website: https://www.java.com/en/download/, download and install the latest version.

Downloading Arduino IDE:

Go to arduino.cc webpage, select your operating system (OS) and download the Arduino IDE



arduino.cc/en/main/software

Download the Arduino IDE

ARDUINO 1.8.10

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software.
This software can be used with any Arduino board.
Refer to the Getting Started page for installation instructions.

Windows Installer, for Windows XP and up
Windows ZIP file for non admin install

Windows app Requires Win 8.1 or 10
Get

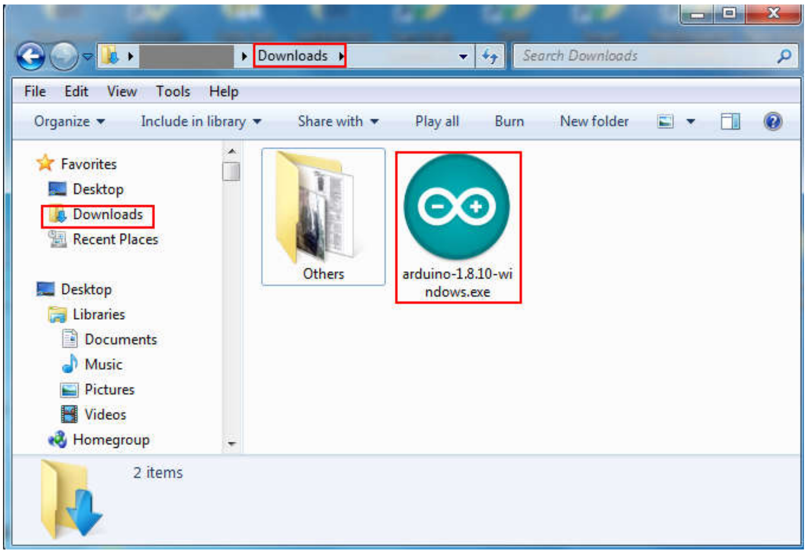
Mac OS X 10.8 Mountain Lion or newer

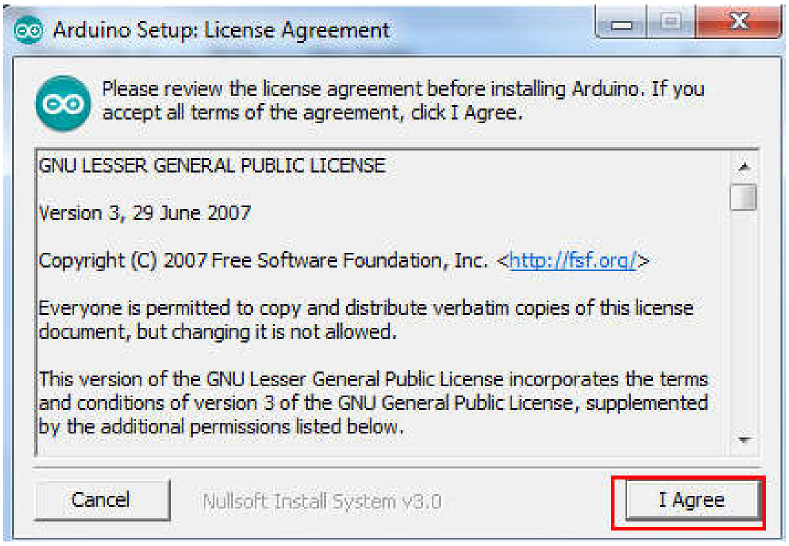
Linux 32 bits
Linux 64 bits
Linux ARM 32 bits
Linux ARM 64 bits

Release Notes
Source Code
Checksums (sha512)

Installing Arduino IDE

Go to your PC **Download** folder and double-click on file named "**arduino-(...).exe**". Follow the installation wizard that shows on your PC screen.




Arduino Setup: License Agreement

Please review the license agreement before installing Arduino. If you accept all terms of the agreement, click I Agree.

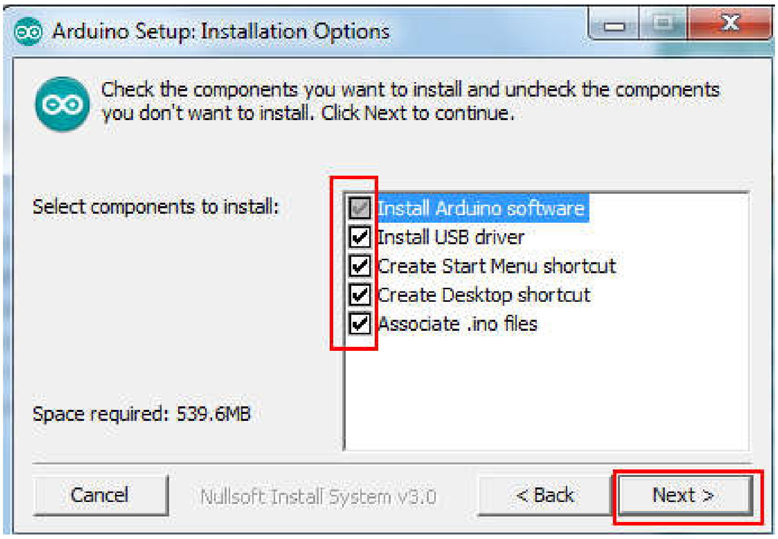
GNU LESSER GENERAL PUBLIC LICENSE

Version 3, 29 June 2007

Copyright (C) 2007 Free Software Foundation, Inc. <http://fsf.org/>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

This version of the GNU Lesser General Public License incorporates the terms and conditions of version 3 of the GNU General Public License, supplemented by the additional permissions listed below.

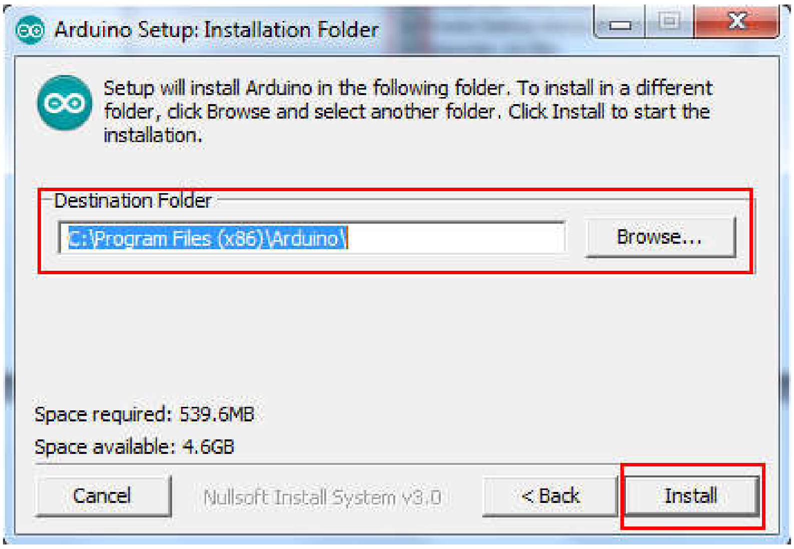

Arduino Setup: Installation Options

Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.

Select components to install:

- ☑ Install Arduino software
- ☑ Install USB driver
- ☑ Create Start Menu shortcut
- ☑ Create Desktop shortcut
- ☑ Associate .ino files

Space required: 539.6MB



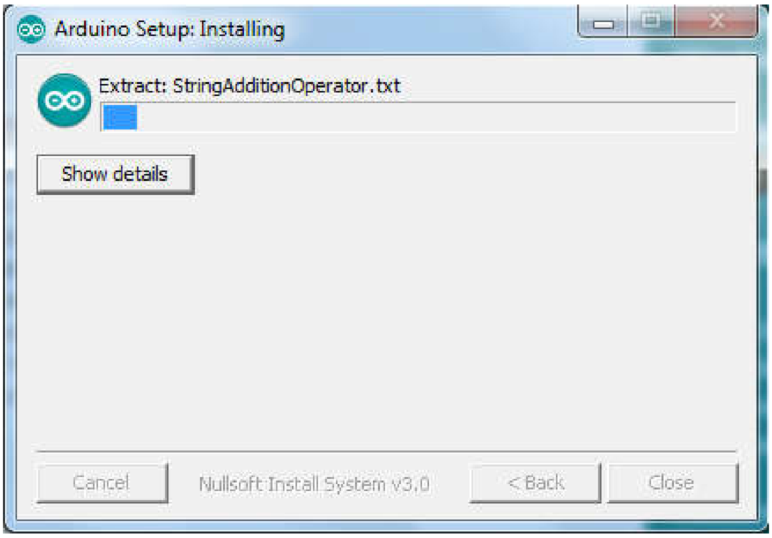
Arduino Setup: Installation Folder

Setup will install Arduino in the following folder. To install in a different folder, click Browse and select another folder. Click Install to start the installation.

Destination Folder: C:\Program Files (x86)\Arduino\

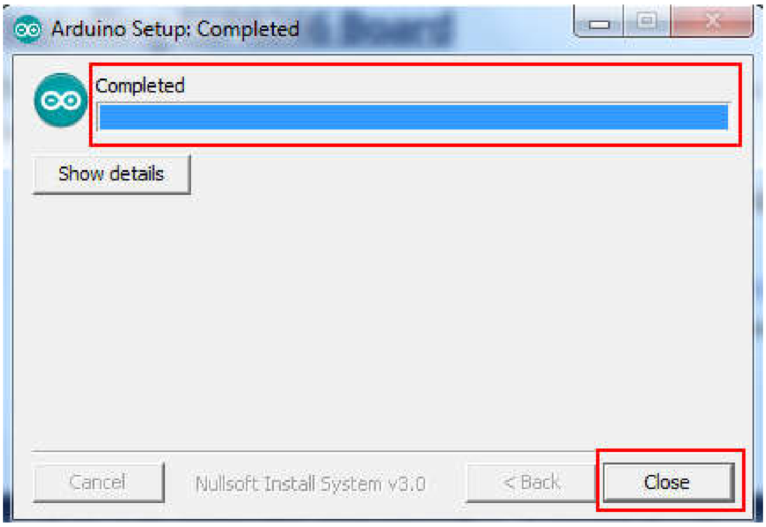
Space required: 539.6MB

Space available: 4.6GB



Arduino Setup: Installing

Extract: StringAdditionOperator.txt



Arduino Setup: Completed

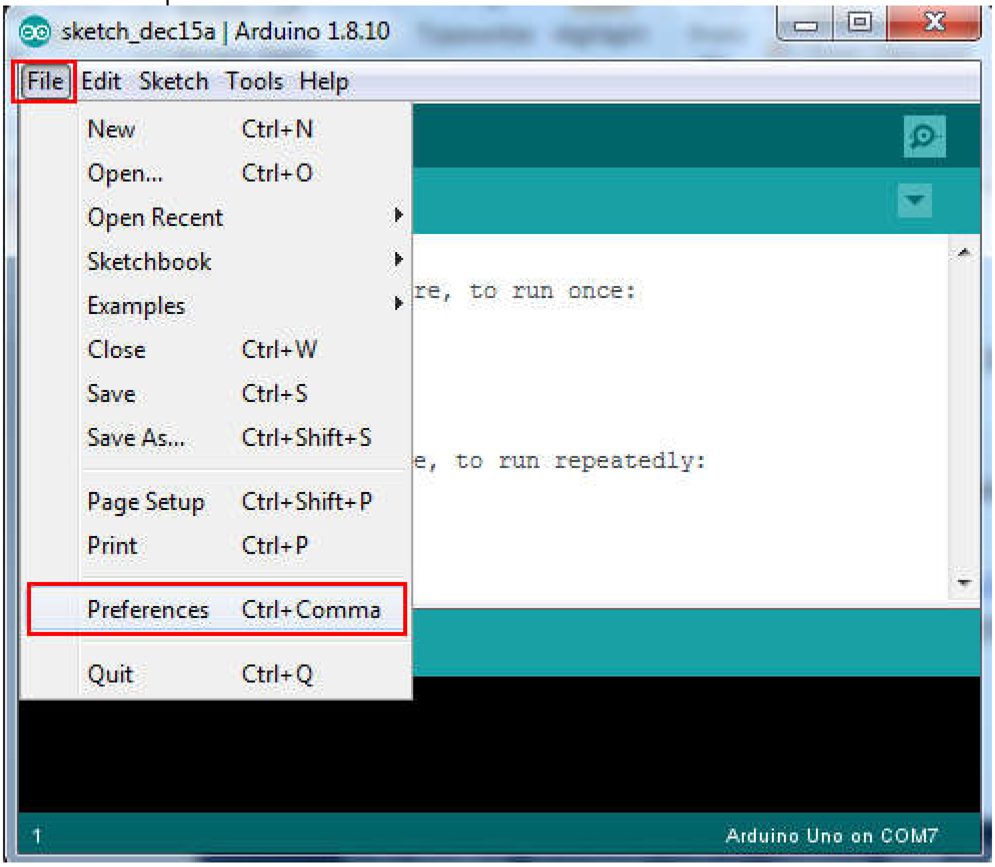
Completed

Congrats! You have installed Arduino IDE to your PC now. The Arduino environment has to be set up to make it compatible with the ESP32 development board. It is required to have the latest Arduino IDE version in order to install the ESP32's platform packages.

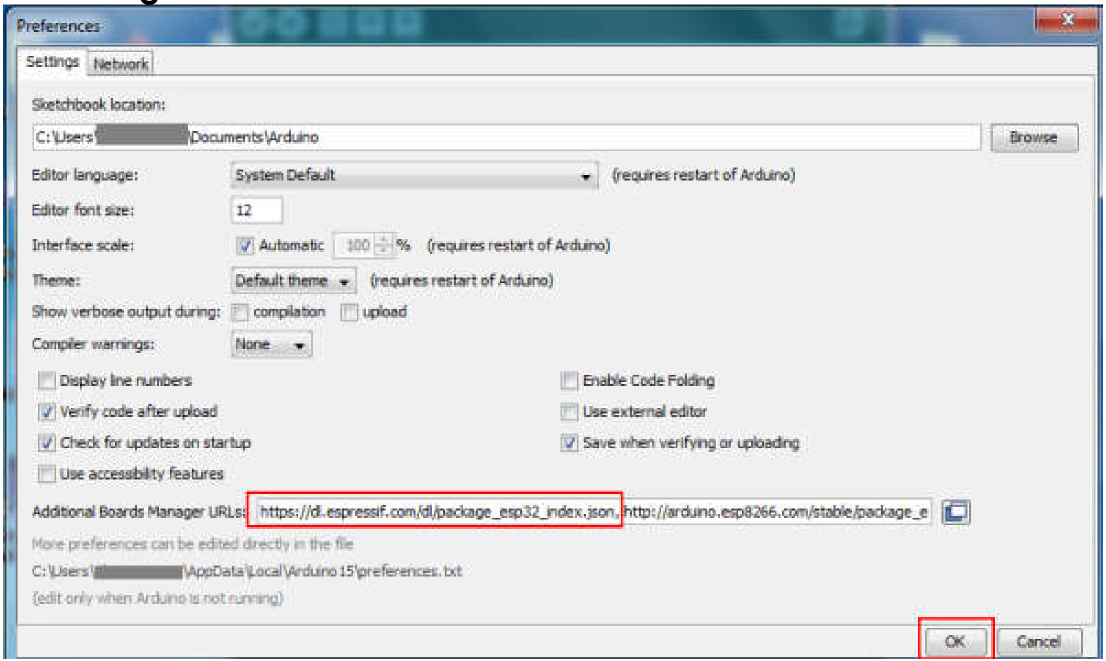
Installing ESP32's platform packages

Double-click on Arduino shortcut, located on your PC desktop.

The Arduino IDE opens. Go to **File -> Preferences.**



Type **https://dl.espressif.com/dl/package_esp32_index.json** into **Additional Board Manager URLs:** field and click on **OK** button.



Sketchbook location:
C:\Users\[redacted]\Documents\Arduino

Editor language: System Default (requires restart of Arduino)

Editor font size: 12

Interface scale: ☑ Automatic 100 % (requires restart of Arduino)

Theme: Default theme (requires restart of Arduino)

Show verbose output during: □ compilation □ upload

Compiler warnings: None

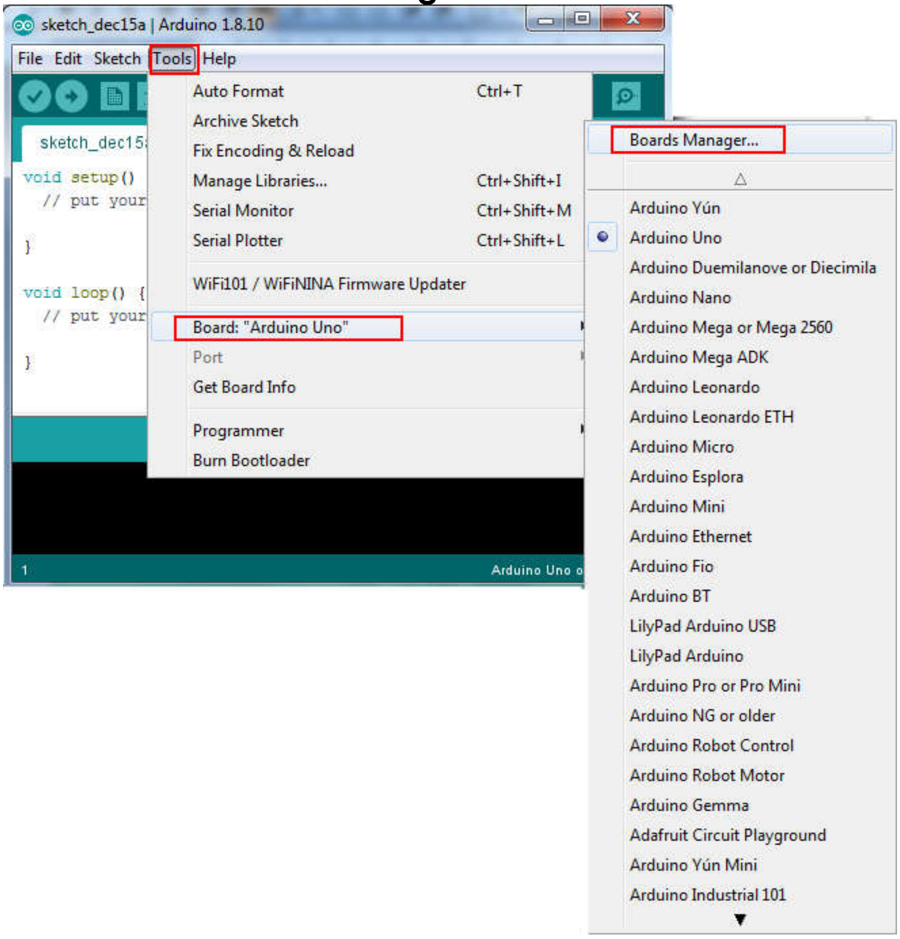
□ Display line numbers
☑ Verify code after upload
☑ Check for updates on startup
□ Use accessibility features
□ Enable Code Folding
□ Use external editor
☑ Save when verifying or uploading

Additional Boards Manager URLs: https://dl.espressif.com/dl/package_esp32_index.json, http://arduino.esp8266.com/stable/package_e

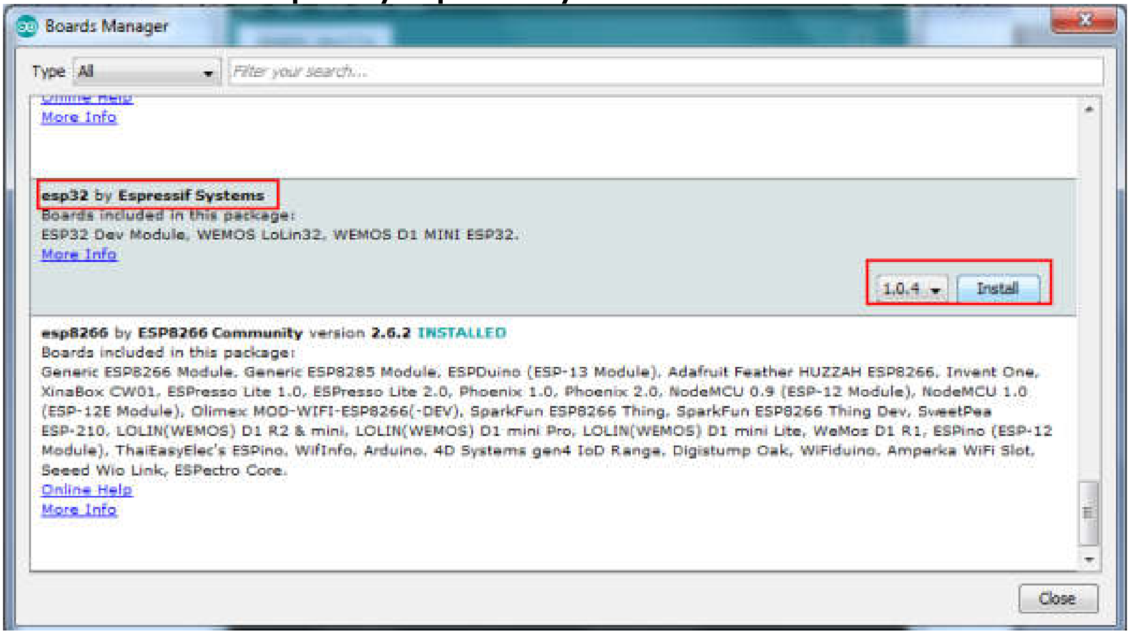
More preferences can be edited directly in the file
C:\Users\[redacted]\AppData\Local\Arduino15\preferences.txt
(edit only when Arduino is not running)

If you already have a URL in there, and want to keep it, you can separate multiple URLs by placing a comma between them. (Arduino IDE 1.6.5 added an expanded text box, separate links in here by line.)

Go to **Tools -> Board -> Boards Manager...**



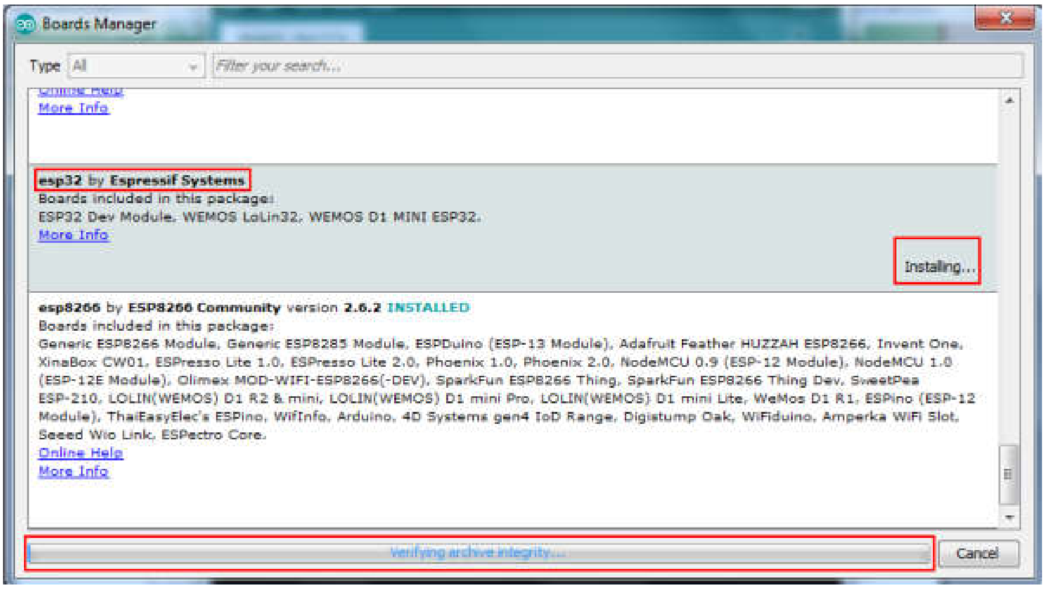
Scroll down, select **esp32 by Espressif Systems** and click on **Install** button.



Online Help
More Info

esp32 by **Espressif Systems**
Boards included in this package:
ESP32 Dev Module, WEMOS LoLin32, WEMOS D1 MINI ESP32.
More Info

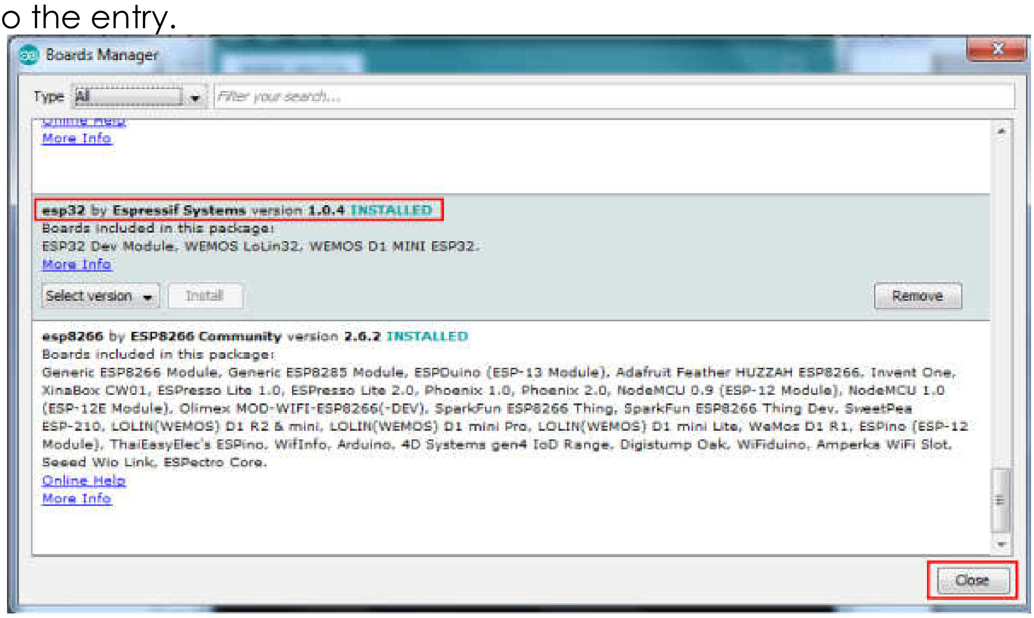
esp8266 by **ESP8266 Community** version **2.6.2** INSTALLED
Boards included in this package:
Generic ESP8266 Module, Generic ESP8285 Module, ESPDuino (ESP-13 Module), Adafruit Feather HUZZAH ESP8266, Invent One, XinaBox CW01, ESPresso Lite 1.0, ESPresso Lite 2.0, Phoenix 1.0, Phoenix 2.0, NodeMCU 0.9 (ESP-12 Module), NodeMCU 1.0 (ESP-12E Module), Olimex MOD-WIFI-ESP8266(-DEV), SparkFun ESP8266 Thing, SparkFun ESP8266 Thing Dev, SweetPea ESP-210, LOLIN(WEMOS) D1 R2 & mini, LOLIN(WEMOS) D1 mini Pro, LOLIN(WEMOS) D1 mini Lite, WeMos D1 R1, ESPino (ESP-12 Module), ThaiEasyElec's ESPino, WifInfo, Arduino, 4D Systems gen4 IoD Range, Digistump Oak, WiFiduino, Amperka WiFi Slot, Seeed Wio Link, ESPectro Core.
Online Help
More Info



esp32 by **Espressif Systems**
Boards included in this package:
ESP32 Dev Module, WEMOS LoLin32, WEMOS D1 MINI ESP32.
More Info

esp8266 by **ESP8266 Community** version **2.6.2** INSTALLED
Boards included in this package:
Generic ESP8266 Module, Generic ESP8285 Module, ESPDuino (ESP-13 Module), Adafruit Feather HUZZAH ESP8266, Invent One, XinaBox CW01, ESPresso Lite 1.0, ESPresso Lite 2.0, Phoenix 1.0, Phoenix 2.0, NodeMCU 0.9 (ESP-12 Module), NodeMCU 1.0 (ESP-12E Module), Olimex MOD-WIFI-ESP8266(-DEV), SparkFun ESP8266 Thing, SparkFun ESP8266 Thing Dev, SweetPea ESP-210, LOLIN(WEMOS) D1 R2 & mini, LOLIN(WEMOS) D1 mini Pro, LOLIN(WEMOS) D1 mini Lite, WeMos D1 R1, ESPino (ESP-12 Module), ThaiEasyElec's ESPino, WifInfo, Arduino, 4D Systems gen4 IoD Range, Digistump Oak, WiFiduino, Amperka WiFi Slot, Seeed Wio Link, ESPectro Core.
Online Help
More Info

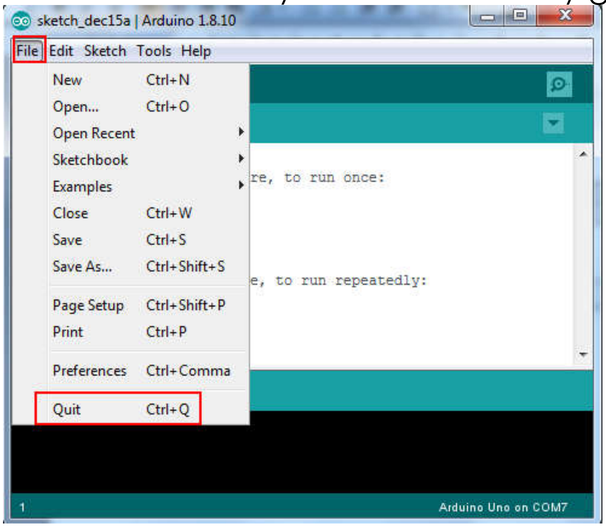
Once the installation has completed, an Arduino-blue "**INSTALLED**" will appear next to the entry.



esp32 by **Espressif Systems** version **1.0.4** INSTALLED
Boards included in this package:
ESP32 Dev Module, WEMOS LoLin32, WEMOS D1 MINI ESP32.
More Info

esp8266 by **ESP8266 Community** version **2.6.2** INSTALLED
Boards included in this package:
Generic ESP8266 Module, Generic ESP8285 Module, ESPDuino (ESP-13 Module), Adafruit Feather HUZZAH ESP8266, Invent One, XinaBox CW01, ESPresso Lite 1.0, ESPresso Lite 2.0, Phoenix 1.0, Phoenix 2.0, NodeMCU 0.9 (ESP-12 Module), NodeMCU 1.0 (ESP-12E Module), Olimex MOD-WIFI-ESP8266(-DEV), SparkFun ESP8266 Thing, SparkFun ESP8266 Thing Dev, SweetPea ESP-210, LOLIN(WEMOS) D1 R2 & mini, LOLIN(WEMOS) D1 mini Pro, LOLIN(WEMOS) D1 mini Lite, WeMos D1 R1, ESPino (ESP-12 Module), ThaiEasyElec's ESPino, WifInfo, Arduino, 4D Systems gen4 IoD Range, Digistump Oak, WiFiduino, Amperka WiFi Slot, Seeed Wio Link, ESPectro Core.
Online Help
More Info

Congrats! You have downloaded, installed Arduino IDE and prepared it for ESP8266. Click on **Close** button.Close your Arduino IDE by goining to **File->Quit**



Do Wiring

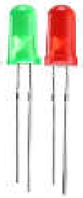
Let's build a standalone ESP32 Web Server that controls two outputs (in our case two LEDs). You can then replace LEDs with any other electronic appliances.

This ESP32 Web Server can be accessed with any device (smartphone, tablet, PC) through any web browser on your local network.

You need the following parts to build your circuit:

ESP32 DOIT DEVKIT V1 Board 1 pc

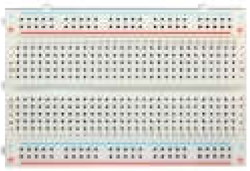
LED 2 pcs



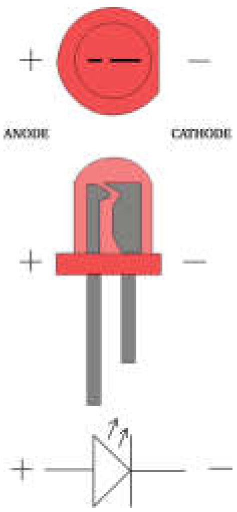
Resistor 2 pcs (220 or 330 Ohms)

Jumper cables (wires) female to male 3 pcs

Breadboard medium size 1 pc



USB A / micro USB B cable1 pc



Please note: LED longer leg is positive (anode), the shorter leg is negative (cathode).

1. Connect green LED anode pin to GPIO 26 pin, red LED anode pin to GPIO 27 pin of your ESP32 development board.
2. Connect green LED and red LED cathode pins through resistors to ground (G) pin of your ESP32 development board.

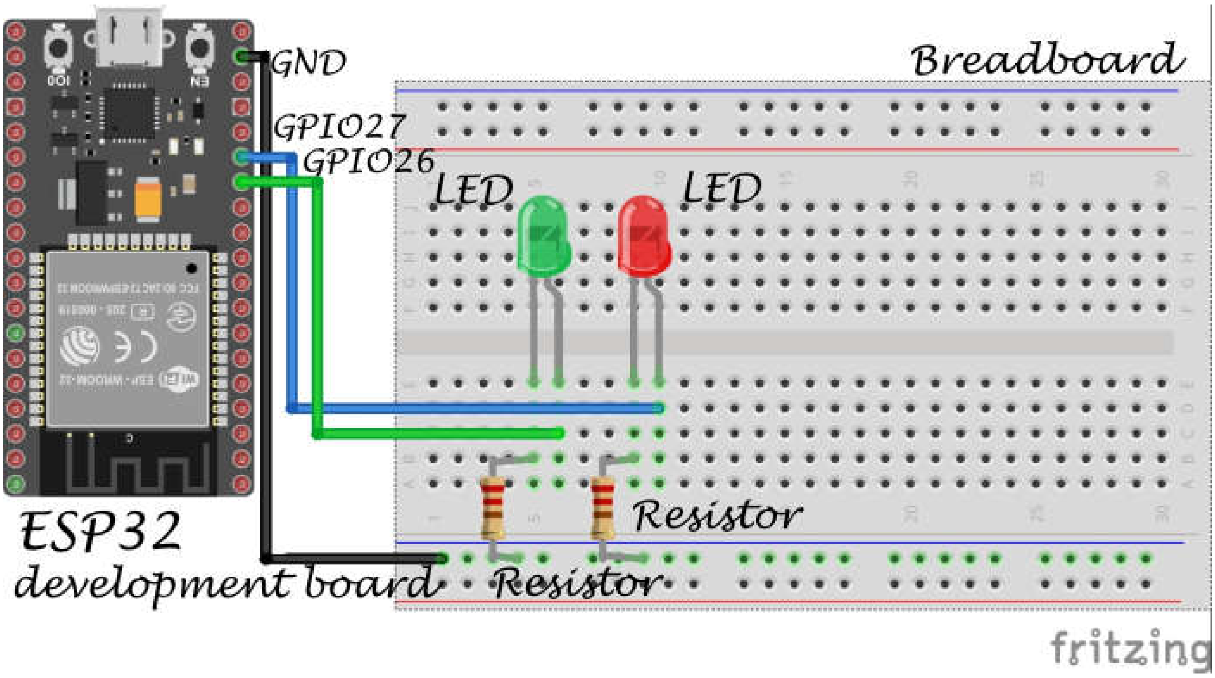


Image credit: Fritzing

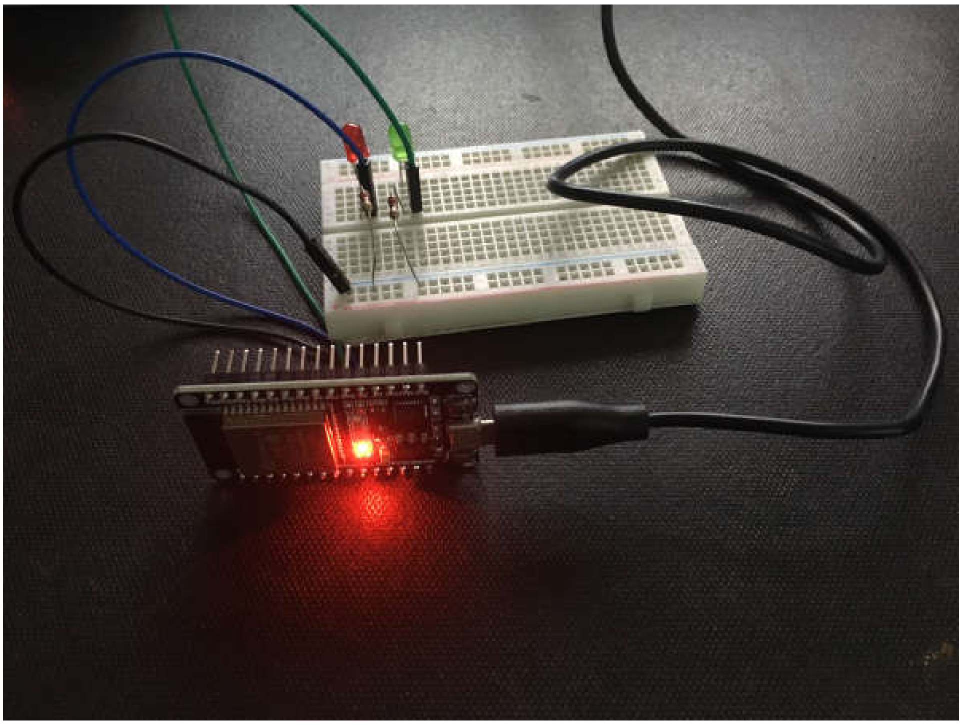


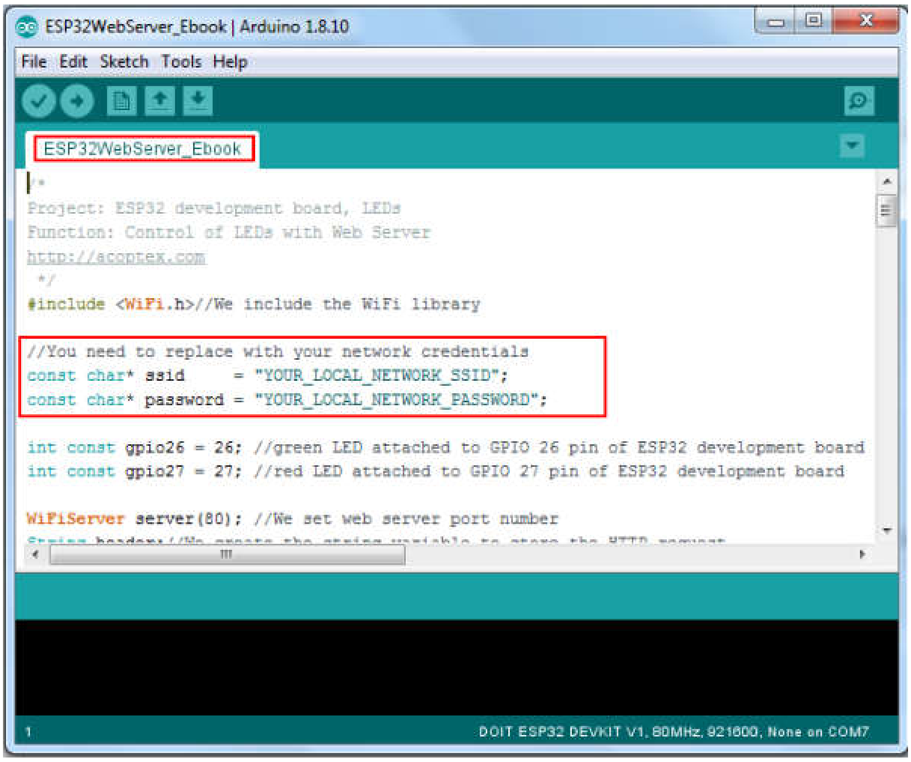
Image credit: Acoptex

Uploading The Sketch

If you are using an ESP32 development board with CP2102 chip, uploading the sketch is very simple, since it has built-in programmer.
Before use ESP32 development board, you need to download the manufacture's driver (CH340) for this chip and install it in your PC - http://www.wch.cn/download/CH341SER_EXE.html . See the description of driver installation package below: CH340/CH341 USB to serial WINDOWS driver installation package that supports 32/64 bit Windows 10 / 8.1 / 8/7 / VISTA / XP, SERVER 2016/2012/2008/2003, 2000 / ME / 98, through Microsoft digital signature authentication, support USB to 3-wire and 9-wire serial port, with the product release To the end user. Applicable scope: CH340G, CH340C, CH340B, CH340E, CH340T, CH340R, CH341A, CH341T, CH341H chips.

If you have CP2102 chip then you need to download the manufacture's driver for this chip and install it in your PC. You can download them here: https://www.silabs.com/products/development-tools/software/usb-to-uart-bridge-vcp-drivers

1. Plug your ESP32 DOIT DEVKIT V1into your PC USB port.
2. Re-open your Arduino IDE.
3.Go to GitHub and download the sketch:
https://github.com/AcoptexCom/Ebooks/blob/master/ESP32WebServer_Ebook/ESP32WebServer_Ebook.ino
4. Open the sketch in Arduino IDE editor.

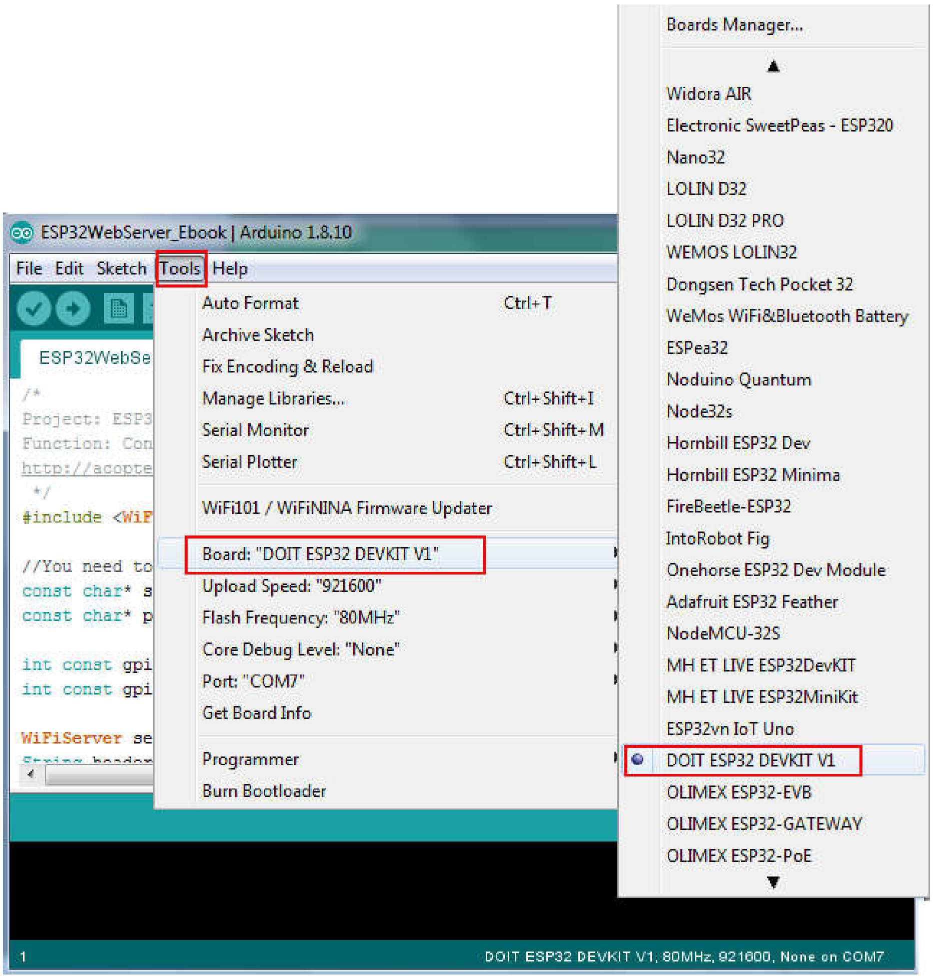

ESP32WebServer_Ebook | Arduino 1.8.10
File Edit Sketch Tools Help
ESP32WebServer_Ebook

```
/*
Project: ESP32 development board, LEDs
Function: Control of LEDs with Web Server
http://acoptex.com
 */
#include <WiFi.h>//We include the WiFi library

//You need to replace with your network credentials
const char* ssid     = "YOUR_LOCAL_NETWORK_SSID";
const char* password = "YOUR_LOCAL_NETWORK_PASSWORD";

int const gpio26 = 26; //green LED attached to GPIO 26 pin of ESP32 development board
int const gpio27 = 27; //red LED attached to GPIO 27 pin of ESP32 development board

WiFiServer server(80); //We set web server port number
```

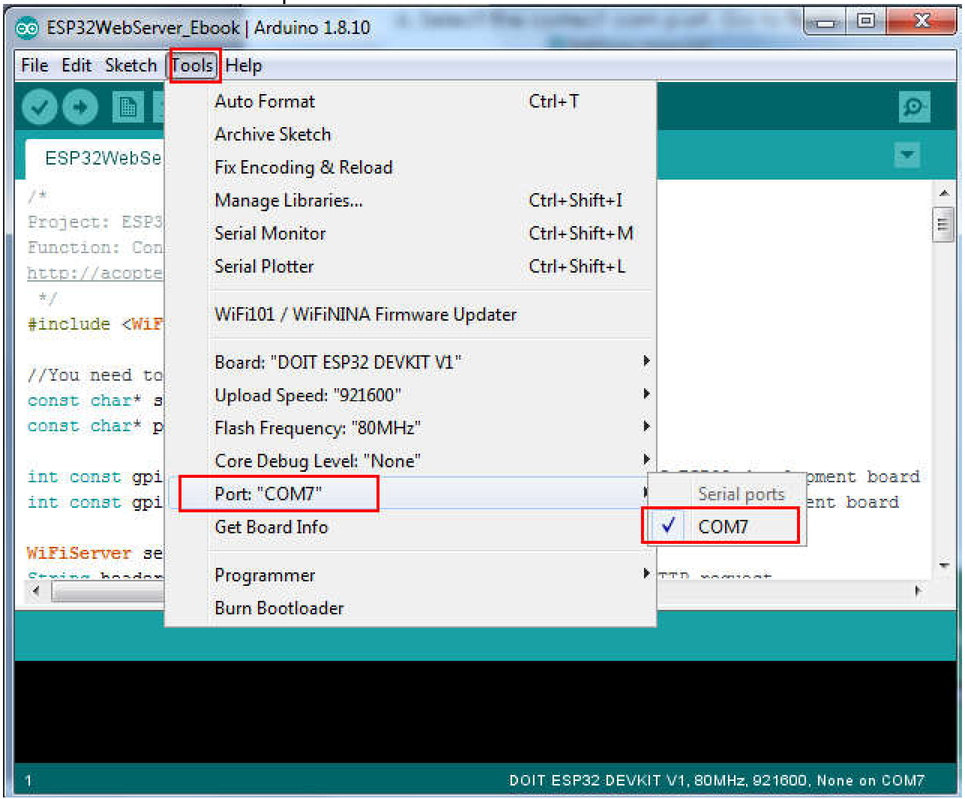
1 DOIT ESP32 DEVKIT V1, 80MHz, 921600, None on COM7

Before uploading the sketch you need to modify the code - type your local network SSID and password.

5. Choose your NodeMCU board. Go to **Tools -> Board -> DOIT ESP32 DEVKIT V1**

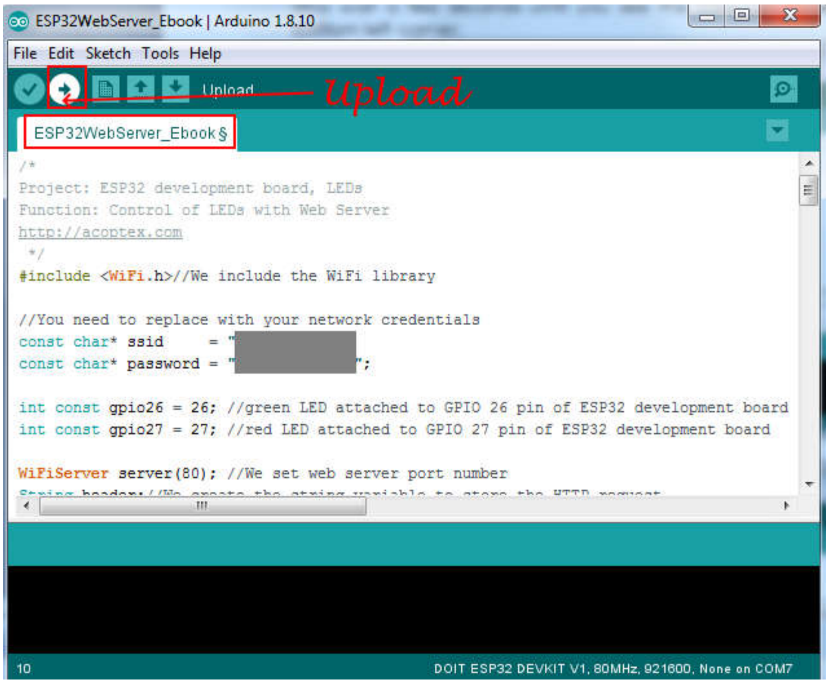


6. Select the correct com port. Go to **Tools -> Port**


ESP32WebServer_Ebook | Arduino 1.8.10
File Edit Sketch Tools Help
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Board: "DOIT ESP32 DEVKIT V1"
Upload Speed: "921600"
Flash Frequency: "80MHz"
Core Debug Level: "None"
Port: "COM7"
Get Board Info
Programmer
Burn Bootloader
Serial ports
✓ COM7
1
DOIT ESP32 DEVKIT V1, 80MHz, 921600, None on COM7

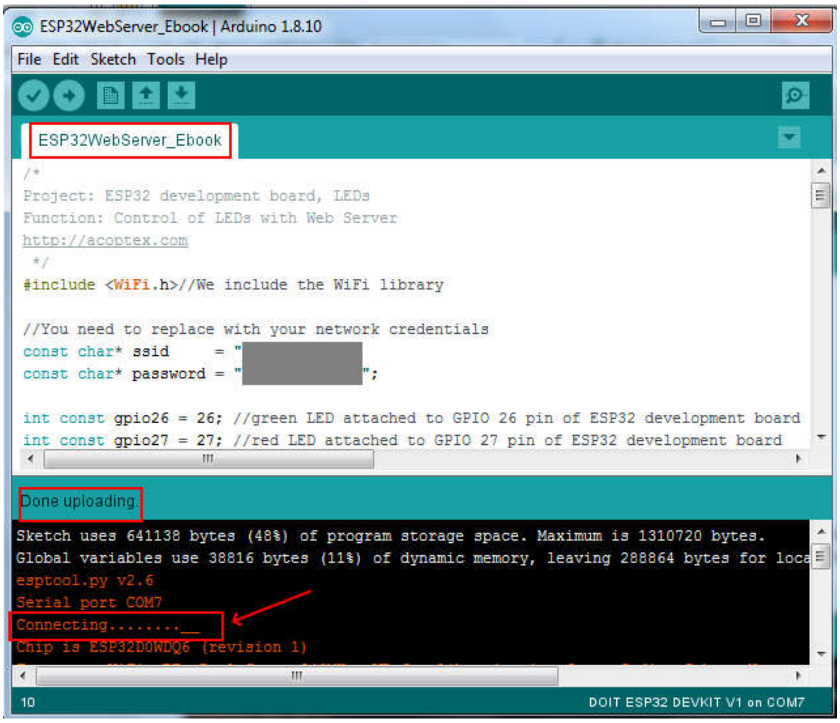
Please note that your COM port is very likely to be different from the preceding screenshot (Port: "COM7"). That is ok, because it doesn't interfere with anything. On the other hand, all the other configurations should look exactly like mine.

7. After checking the configurations, click on **Upload** button in the Arduino IDE and wait a few seconds until you see the message **Done uploading** in the bottom left corner. When you see **Connecting....** press and release the on-board **BOOT** button.


ESP32WebServer_Ebook | Arduino 1.8.10
File Edit Sketch Tools Help
Upload
Upload
ESP32WebServer_Ebook §

```
/*
Project: ESP32 development board, LEDs
Function: Control of LEDs with Web Server
http://acoptex.com
 */
#include <WiFi.h>//We include the WiFi library

//You need to replace with your network credentials
const char* ssid     = "[redacted]
const char* password = "[redacted]";

int const gpio26 = 26; //green LED attached to GPIO 26 pin of ESP32 development board
int const gpio27 = 27; //red LED attached to GPIO 27 pin of ESP32 development board

WiFiServer server(80); //We set web server port number
```

10 DOIT ESP32 DEVKIT V1, 80MHz, 921600, None on COM7


ESP32WebServer_Ebook | Arduino 1.8.10
File Edit Sketch Tools Help
ESP32WebServer_Ebook

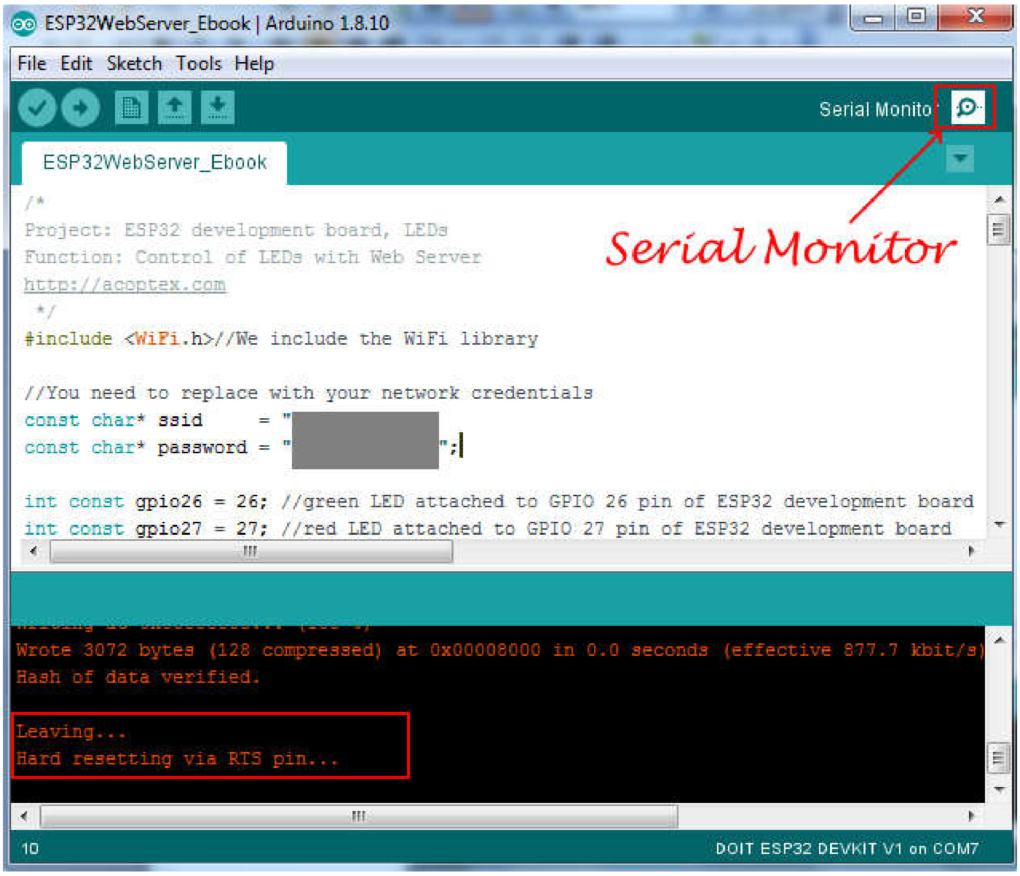
```
/*
Project: ESP32 development board, LEDs
Function: Control of LEDs with Web Server
http://acoptex.com
 */
#include <WiFi.h>//We include the WiFi library

//You need to replace with your network credentials
const char* ssid     = "[redacted]
const char* password = "[redacted]";

int const gpio26 = 26; //green LED attached to GPIO 26 pin of ESP32 development board
int const gpio27 = 27; //red LED attached to GPIO 27 pin of ESP32 development board
```

Done uploading.

```
Sketch uses 641138 bytes (48%) of program storage space. Maximum is 1310720 bytes.
Global variables use 38816 bytes (11%) of dynamic memory, leaving 288864 bytes for loca
esptool.py v2.6
Serial port COM7
Connecting........__
Chip is ESP32D0WDQ6 (revision 1)
```

10 DOIT ESP32 DEVKIT V1 on COM7

8. Click on **Serial Monitor** button and open Serial Monitor at 115200 bps.

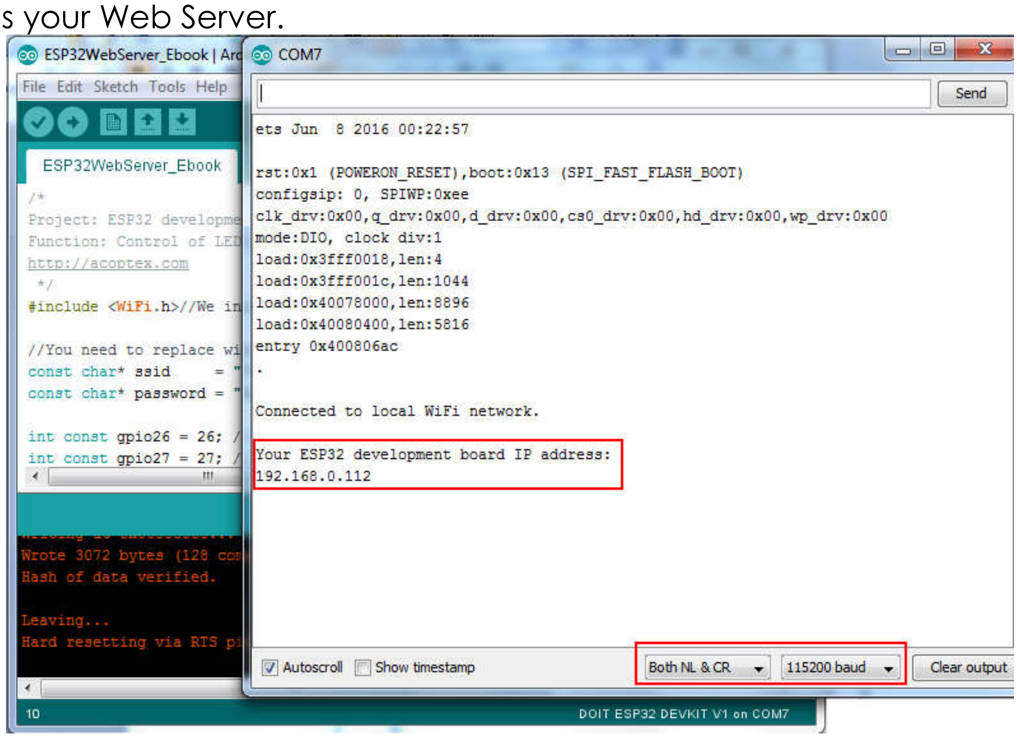

ESP32WebServer_Ebook | Arduino 1.8.10
File Edit Sketch Tools Help
Serial Monito
ESP32WebServer_Ebook
Serial Monitor

```
/*
Project: ESP32 development board, LEDs
Function: Control of LEDs with Web Server
http://acoptex.com
 */
#include <WiFi.h>//We include the WiFi library

//You need to replace with your network credentials
const char* ssid     = "
const char* password = "[redacted]";

int const gpio26 = 26; //green LED attached to GPIO 26 pin of ESP32 development board
int const gpio27 = 27; //red LED attached to GPIO 27 pin of ESP32 development board
```

```
Wrote 3072 bytes (128 compressed) at 0x00008000 in 0.0 seconds (effective 877.7 kbit/s)
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
```

10   DOIT ESP32 DEVKIT V1 on COM7

9. Press on-board **EN** button on ESP32 development board to restart the module. You will see IP address of your ESP32 development board (for example, I have **192.168.0.112**). Copy this IP address, you will need it to access your Web Server.

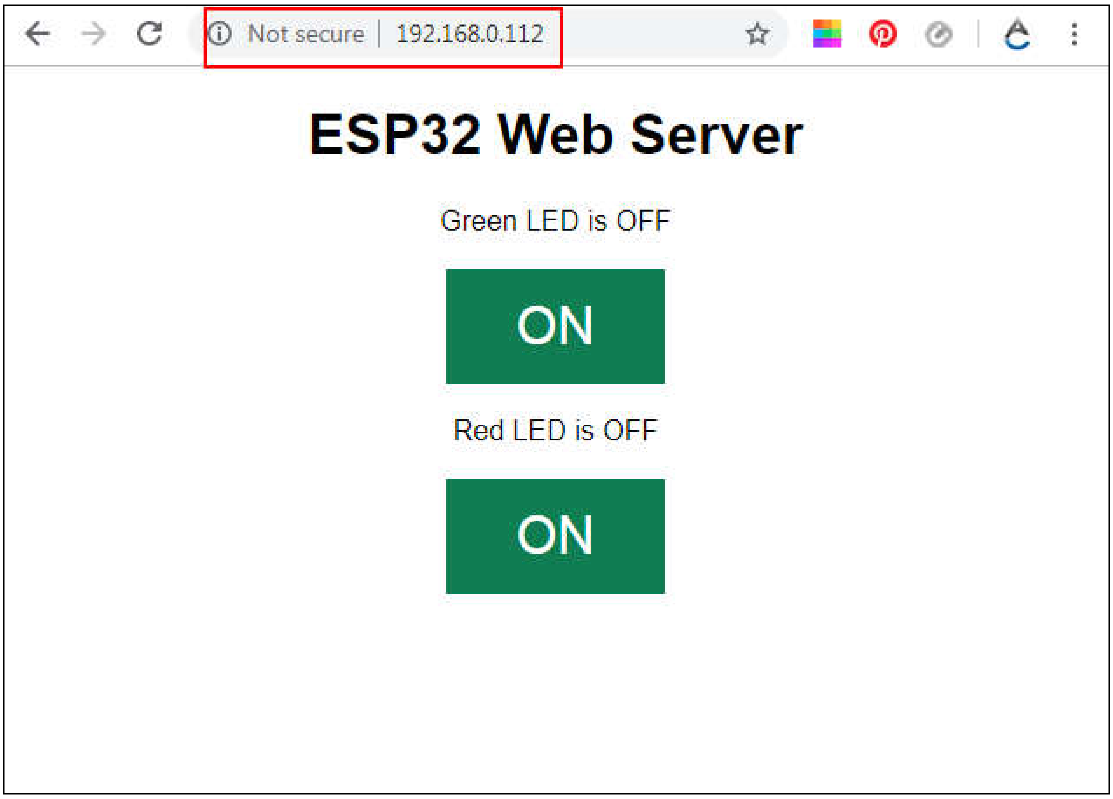


```
ets Jun  8 2016 00:22:57

rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff0018,len:4
load:0x3fff001c,len:1044
load:0x40078000,len:8896
load:0x40080400,len:5816
entry 0x400806ac
.

Connected to local WiFi network.

Your ESP32 development board IP address:
192.168.0.112
```

☑ Autoscroll □ Show timestamp | Both NL & CR | 115200 baud | Clear output

10   DOIT ESP32 DEVKIT V1 on COM7

10. Open any web browser (Chrome, Opera, IE...), type the IP address (I have **192.168.0.112**), and you'll see the following page. This page is sent by the ESP32 development board when you make a request on the IP address.

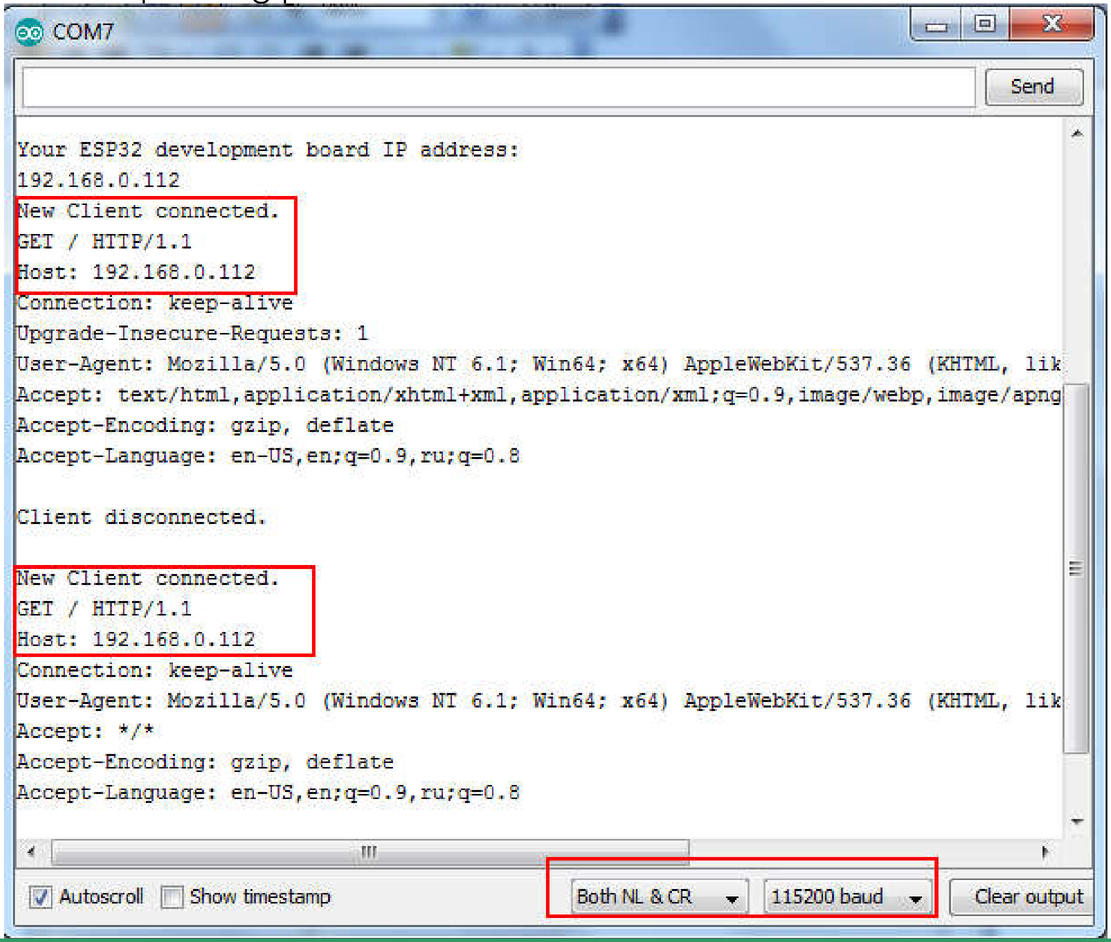


ESP32 Web Server

Green LED is OFF

Red LED is OFF

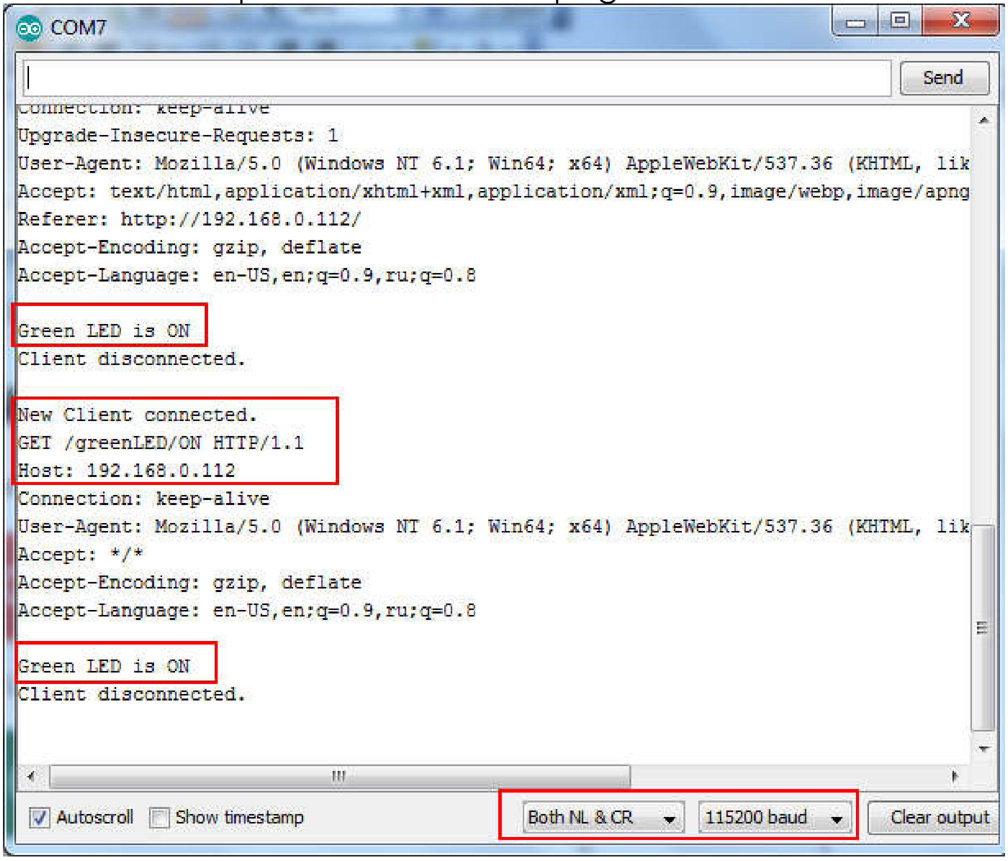
If you take a look at the Serial Monitor, you can see that the ESP32 development board receives an HTTP request from a new client (your web browser). You can also see other information - HTTP header fields, which define the operating parameters of the HTTP transaction.



```
Your ESP32 development board IP address:
192.168.0.112
New Client connected.
GET / HTTP/1.1
Host: 192.168.0.112
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, lik
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9,ru;q=0.8

Client disconnected.

New Client connected.
GET / HTTP/1.1
Host: 192.168.0.112
Connection: keep-alive
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, lik
Accept: */*
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9,ru;q=0.8
```

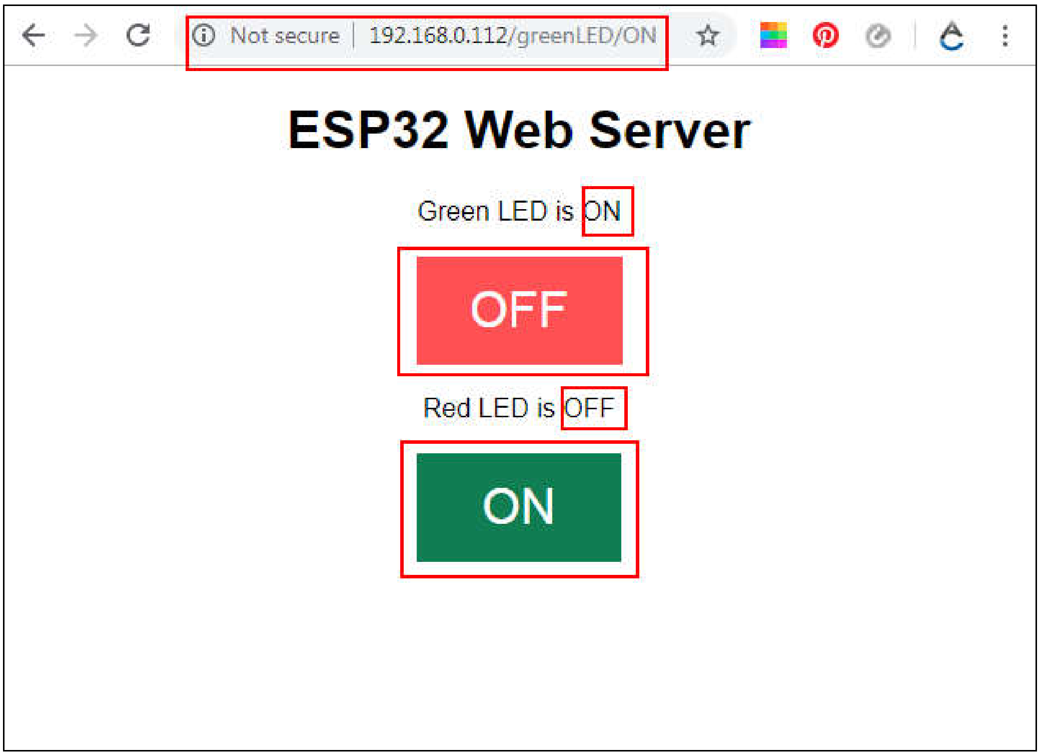
☑ Autoscroll □ Show timestamp Both NL & CR 115200 baud

11. Let's click on ON button to turn green LED ON. The ESP32 development board receives a request on the /greenLED/ON URL, and turns green LED ON. The LED state will be updated on the web page.



```
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, lik
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng
Referer: http://192.168.0.112/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9,ru;q=0.8

Green LED is ON
Client disconnected.

New Client connected.
GET /greenLED/ON HTTP/1.1
Host: 192.168.0.112
Connection: keep-alive
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, lik
Accept: */*
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9,ru;q=0.8

Green LED is ON
Client disconnected.
```

☑ Autoscroll □ Show timestamp Both NL & CR 115200 baud



ESP32 Web Server

Green LED is ON

Red LED is OFF

12. You can also test red LED button and check that it works same way.

The Code

```
#include <WiFi.h>//We include the WiFi library

//You need to replace with your network credentials
const char* ssid     = "YOUR_LOCAL_NETWORK_SSID";
const char* password = "YOUR_LOCAL_NETWORK_PASSWORD";

int const gpio26 = 26; //green LED attached to GPIO 26 pin of ESP32
int const gpio27 = 27; //red LED attached to GPIO 27 pin of ESP32

WiFiServer server(80); //We set web server port number
String header;//We create the string variable to store the HTTP request

//We create variables to store the current output state
String gpio26State = "OFF";
String gpio27State = "OFF";

//Other variables
unsigned long currentTime = millis();
unsigned long previousTime = 0;

const long timeoutTime =2000; //timeout time in milliseconds (2 seconds)

void setup() {//The function only runs once when your ESP32 boots.
  Serial.begin(115200); //Initialise serial communication at 115200 bps
  pinMode(gpio26, OUTPUT);//We set gpio26 pin as OUTPUT
  pinMode(gpio27, OUTPUT);//We set gpio27 pin as OUTPUT
  digitalWrite(gpio26, LOW);//We set gpio 26 pin to LOW by default
  digitalWrite(gpio27, LOW);//We set gpio 27 pin to LOW by default

  //We connect to local WiFi network
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  //We print in Serial Monitor ESP32 IP address
  Serial.println("");
  Serial.println("");
  Serial.println("Connected to local WiFi network.");
  Serial.println("");
  Serial.println("Your ESP32 development board IP address: ");
  Serial.println(WiFi.localIP());

  server.begin();//We start the web server
}

void loop(){//We define what happens when a new client establishes a
connection with the web server.
  WiFiClient client = server.available();   //We listen for incoming clients
/*When a request is received from a client, we will save the incoming data. The
while loop that follows will be running as long as the client stays connected.*/
  if (client) {                             //If a new client connects,
    Serial.println("New Client connected.");//We print a message out in the Serial
Monitor
    String currentLine = "";                //We make a String variable to hold
incoming data from the client
    currentTime = millis();
    previousTime = currentTime;
    while (client.connected() && currentTime - previousTime <= timeoutTime) { //We
have loop while the client's connected
      currentTime = millis();
      if (client.available()) {//If there are bytes to read from the client,
        char c = client.read();//Read a byte, then
        Serial.write(c);      //Print it out the Serial Monitor
        header += c;
        if (c == '\n') {      //If the byte is a newline character
//If the current line is blank, you got two newline characters in a row.
//That's the end of the client HTTP request, so send a response:
          if (currentLine.length() == 0) {
            client.println("HTTP/1.1 200 OK"); //HTTP headers always start with a
response code
            client.println("Content-type:text/html");//and a content-type so the
client knows what's coming,
            client.println("Connection: close");
            client.println();                        //then a blank line
//The next section of if and else statements checks which button was pressed
in your web page, and controls the outputs accordingly
 //We make a request on different URLs depending on the button we click on
            if (header.indexOf("GET /greenLED/ON") >= 0) { //turns the LEDs ON/OFF
```

```
              Serial.println("Green LED is ON");
              Gpio26State = "ON";
              digitalWrite(gpio26, HIGH);
            } else if (header.indexOf("GET /greenLED/OFF") >= 0) {
              Serial.println("Green LED is OFF");
              Gpio26State = "OFF";
              digitalWrite(gpio26, LOW);
            } else if (header.indexOf("GET /redLED/ON") >= 0) {
              Serial.println("Red LED is ON");
              Gpio27State = "ON";
              digitalWrite(gpio27, HIGH);
            } else if (header.indexOf("GET /redLED/OFF") >= 0) {
              Serial.println("Red LED is OFF");
              Gpio27State = "OFF";
              digitalWrite(gpio27, LOW);
            }
/*For example, if you have pressed the green LED ON button, the URL changes to
the ESP32 development board IP address followed by /greenLED/ON, and we receive
that information on the HTTP header. If it contains GET /greenLED/ON, the code
prints a message in the Serial Monitor, changes the gpio26State variable to ON,
and turns the LED ON. It is the same for the other buttons. If you want to add
more outputs, you should modify this part of the code to include them*/
 //Then we make the HTML web page
    client.println("<!DOCTYPE html><html>"); //Indicates that we are sending HTML
     client.println("<head><meta name=\"viewport\" content=\"width=device-width,
initial-scale=1\">");//It makes the web page responsive in any web browser
            client.println("<link rel=\"icon\" href=\"data:,\">"); //We prevent
requests related to the favicon
            //CSS style for ON/OFF buttons
            //You can change the background-color, font-size, make border, change
font color if you want so
            client.println("<style>html { font-family: Helvetica; display: inline-
block; margin: 0px auto; text-align: center;}");
            client.println(".button { background-color: #2E7C4F; border: none;
color: white; padding: 16px 40px;");  //ON button CSS style
            client.println("text-decoration: none; font-size: 30px; margin:
2px;}");
            client.println(".button1 {background-color: #FF4C4F; border: none;
color: white; padding: 16px 33px;");  //OFF button CSS style
            client.println("text-decoration: none; font-size: 30px; margin:
2px;}</style></head>");
            client.println("<body><h1>ESP32 Web Server</h1>");
```

```
//Web page heading
client.println("<p>Green LED is " + gpio26State + "</p>");
//We show the current state, ON/OFF buttons for green LED
if (gpio26State=="ON") {    //If the gpio26State is OFF, it displays the ON button
              client.println("<p><a href=\"/greenLED/OFF\"><button
class=\"button1\">OFF</button></a></p>");

            } else {
              client.println("<p><a href=\"/greenLED/ON\"><button
class=\"button\">ON</button></a></p>");
            }
            client.println("<p>Red LED is " + gpio27State + "</p>");
//We show the current state, ON/OFF buttons for red LED

            if (gpio27State=="ON") {
//If the gpio27State is OFF, it displays the ON button
              client.println("<p><a href=\"/redLED/OFF\"><button
class=\"button1\">OFF</button></a></p>");
            } else {
              client.println("<p><a href=\"/redLED/ON\"><button
class=\"button\">ON</button></a></p>");
            }
            client.println("</body></html>");
            client.println();              //We add the blank line on the end of
the HTTP response

            break;                         //We break out of the while loop
          } else {                         //If you got a newline, then clear
currentLine
            currentLine = "";
          }
        } else if (c != '\r') {            //If you got anything else but a
carriage return character,
          currentLine += c;                //add it to the end of the currentLine
        }
      }
    }

    header = "";                           //We clear the header variable
    client.stop();                         //We close the connection
    Serial.println("Client disconnected.");//We print the message in Serial
Monitor
```

```
 Serial.println();
  }
}
```

The Source code is published on GitHub:
https://github.com/AcoptexCom/Ebooks/blob/master/ESP32WebServer_Ebook/ESP32WebServer_Ebook.ino