ESP32 WiFi IoT Development Board Example Guide

Table of Contents

Contents

Table of Contents 2

Introduction 3

Tutorial Content 3

Parts Needed 3

Hardware Overview 4

Software Overview 6

Section 1: Setup 6

1.1 Hardware Setup 6

1.2 Software setup: install USB to serial chip driver 7

1.3 Software Setup install ESP32 Arduino core files 8

Section 2: Running with Arduino 9

Example 2.1 ESP32 Blink 9

Example 2.2 ESP32 WiFi Scan 11

Section 3: WiFi and Internet communication example 11

3.1 Example: WiFi local Webserver 12

3.2 Example: Fetch HTML content of a live website 17

Going Further 20

Other Examples 21

Introduction

The **ESP32 WiFi IoT Development Board** is based on the latest ESP32 chip by Espressif. At a glance the ESP32 is a big brother to the earlier chip ESP8266. The ESP32 boost a 2x to 3x improvement to the capabilities and features of the ESP8266. The ESP32 is dual core that runs at maximum 240MHz, has 4MB flash memory, supports 10x capacitive touch interface, more UARTS, SPI, i2c and most especially now supports the BLE (Bluetooth Low Energy) technology. You can do a lot of WiFi and IoT projects that you've done form the previous chip and now even capable of extending it to more powerful application.

This development board is ready to use. Simply plug a micro-USB cable in the USB port and you can run Arduino software and even the advanced ESP-IDF application. It has an onboard USB-to-serial circuit so you can transfer Arduino code immediately. The board also has a Li-ion battery supply and charging port for modular application and battery operations.

(Note: This board is based on Adafruit Feather32 Dev Board and TTGO/LOLIN/WEMOS ESP32 Dev Board)

Tutorial Content

In this Example guide we will be exploring the following topics:

- ESP32 chip Hardware and Software Overview
- ESP32 and Arduino Setup and preparing the development tools
- Run Examples from Arduino with Blink, WiFi and more

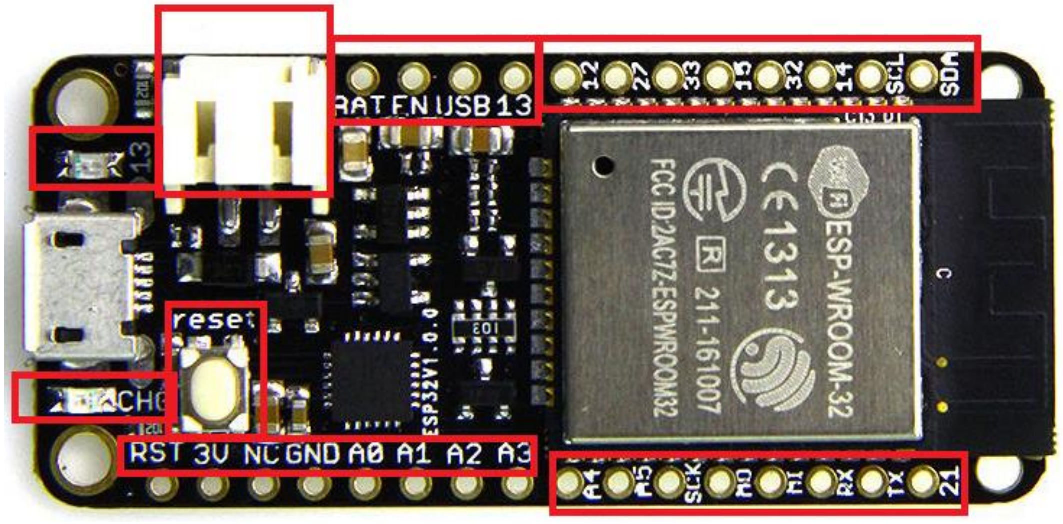
Parts Needed

We will need the following items:

- ESP32 WiFi IoT Development Board
- USB Cable – USB A to micro USB
- Breadboard half size or full-size
- LED (any color)
- 330 ohm resistor
- Connecting wire male to male this or this premium male to male

Hardware Overview

The **ESP32 WiFi IoT Development board** is based on the **ESP32** chip by Espressif. On the board is a variant of the ESP32 chip, the **ESP32-WROOM32** which houses the esp32 chip inside and includes the necessary external components to run the esp32 chip correctly such as RAM chip, RF shielding and pin outing.



Pin functions

Below are the pins descriptions and informations for the Development board.

GND – this is the common ground for all power and logic
BAT – this is the positive voltage to/from the JST jack for the optional Lipoly battery
USB – this is the positive voltage to/from the micro USB jack if connected
EN – this is the 3.3V regulator's enable pin. It's pulled up, so connect to ground to disable the 3.3V regulator
3V – this is the output from the 3.3V regulator. The regulator can supply 500mA peak but half of that is drawn by the ESP32, and it's a fairly power-hungry chip. So if you need a ton of power for stuff like LEDs, motors, etc. Use the USB or BAT pins, and an additional regulator
GND – this is the common ground for all power and logic
BAT – this is the positive voltage to/from the JST jack for the optional Lipoly battery

USB – this is the positive voltage to/from the micro USB jack if connected

GPIO & Analog Pins:

There are many GPIO and analog inputs available to you for connecting LEDs, buttons, switches, sensors, etc. Here's the pin descriptions.

Bottom row:
A0 – this is an analog input A0 and also an analog output DAC2. It can also be used as a GPIO #26.
A1 – this is an analog input A1 and also an analog output DAC1. It can also be used as a GPIO #25.
A2 – this is an analog input A2 and also GPI #34. Note it is not an output-capable pin!
A3 – this is an analog input A3 and also GPI #39. Note it is not an output-capable pin!
A4 – this is an analog input A4 and also GPIO #36.
A5 – this is an analog input A5 and also GPIO #4.
21 – General purpose IO pin #21.

Top row:
13 – This is GPIO #13 and also an analog input A12. It's also connected to the red LED next to the USB port. It's also connected to the red LED next to the USB port.
12 – This is GPIO #12 and also an analog input A11. This pin has a pull-down resistor built into it, we recommend using it as an output only, or making sure that the pull-down is not affected during boot.
27 – This is GPIO #27 and also an analog input A10
33 – This is GPIO #33 and also an analog input A9. It can also be used to connect a 32 KHz crystal.
15 – This is GPIO #15 and also an analog input A8
32 – This is GPIO #32 and also an analog input A7. It can also be used to connect a 32 KHz crystal.
14 – This is GPIO #15 and also an analog input A6

Logic pins:

This is the general purpose I/O pin set for the microcontroller. All logic is 3.3V
The ESP32 runs on 3.3V power and logic, and unless otherwise specified, GPIO pins are not 5V safe!

Serial pins:

RX and TX are the additional Serial1 pins, and are not connected to the USB/Serial converter. That means you can use them to connect to UART-devices like GPS's, fingerprint sensors, etc. The TX pin is the output from the module. The RX pin is the input into the module. Both are 3.3V logic.

I2C & SPI pins:

You can use the ESP32 to control I2C and SPI devices, sensors, outputs, etc. If using with Arduino, the standard **Wire** and **SPI** devices work as is. Note that the I2C pins do not have pullup resistors already! You must add them if you want to communicate with an I2C device.

Features & Specifications

240 MHz dual core Tensilica LX6 microcontroller with 600 DMIPS
Integrated 520 KB SRAM

Integrated 802.11b/g/n HT40 Wi-Fi transceiver, baseband, stack and LWIP
Integrated dual mode Bluetooth (classic and BLE)
4 MByte flash
On-board PCB antenna
Ultra-low noise analog amplifier
Hall sensor
10x capacitive touch interface
32 kHz crystal oscillator
3 x UARTs (only two are configured by default in the Feather Arduino IDE support, one UART is used for bootloading/debug)
2 x I2C (only one is configured by default in the Feather Arduino IDE support)
12 x ADC input channels
2 x I2S Audio
2 x DAC
PWM/timer input/output available on every GPIO pin
OpenOCD debug interface with 32 kB TRAX buffer
SDIO master/slave 50 MHz
SD-card interface support

If you want to know more about the ESP32 and Espressif company you may go here: ESP32 Overview .

Software Overview

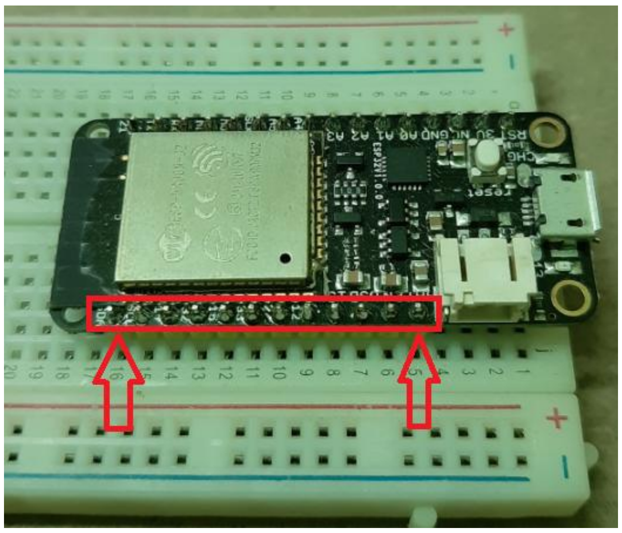
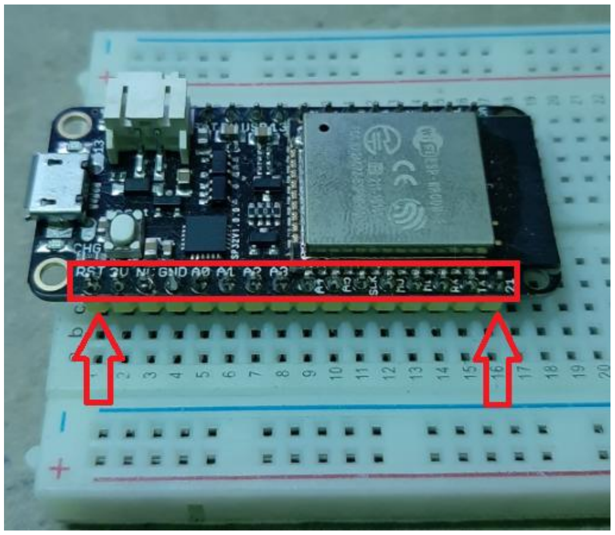
The best way to use the ESP32 is with using Arduino IDE software. You may explore advanced programming by trying also the ESP-IDF development tool (for advanced users or want to explore). The ESP32 is now widely supported by the Arduino environment and is also on continuous development. To start download the latest Arduino software here.

Section 1: Setup

Here we will setup the necessary hardware and software configurations.

1.1 Hardware Setup

The ESP32 Development board may come with the male headers unsoldered (you may have it soldered in advance too). To solder the headers onto the board insert them onto a breadboard following the alignment of the holes on the development board PCB. Align the holes and the headers and using a soldering iron and soldering lead, solder all the pins on both sides.

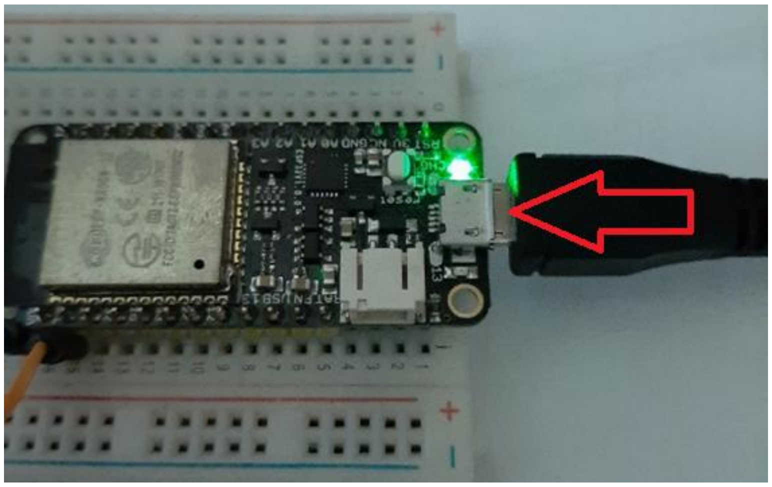
You may keep the esp32 board attached to the breadboard.

1.2 Software setup: install USB to serial chip driver

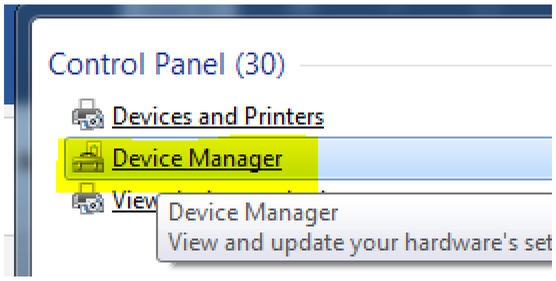
Let's install the software driver for our onboard USB to serial chip based on Silicon Labs CP210XX chip. This chip communicates from the computer to the ESP32 chip to transfer the code from the Arduino IDE to the ESP32.

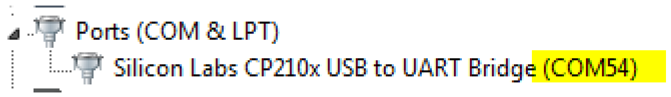
Download the software driver here: SiLabs CP2104 Driver. Extract and install the software driver according to your computer Operating system.

After installing, connect your ESP32 board to your computer with a **Micro USB Cable**.



Go to your **Windows Device Manager**, check for Ports and take note of the **COMX** number.





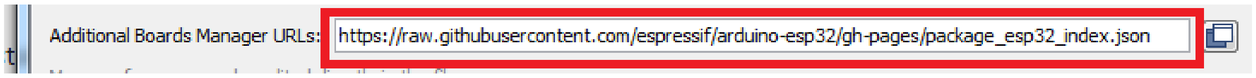
Now your ESP32 Development board is recognized by your computer.

1.3 Software Setup install ESP32 Arduino core files

Now let’s setup the ESP32 board to be used in the Arduino IDE.

Open your Arduino IDE. Go to **File>Preferences**. At the bottom find **Additional Boards Manager URLs**, beside is a blank text box and enter the following text:

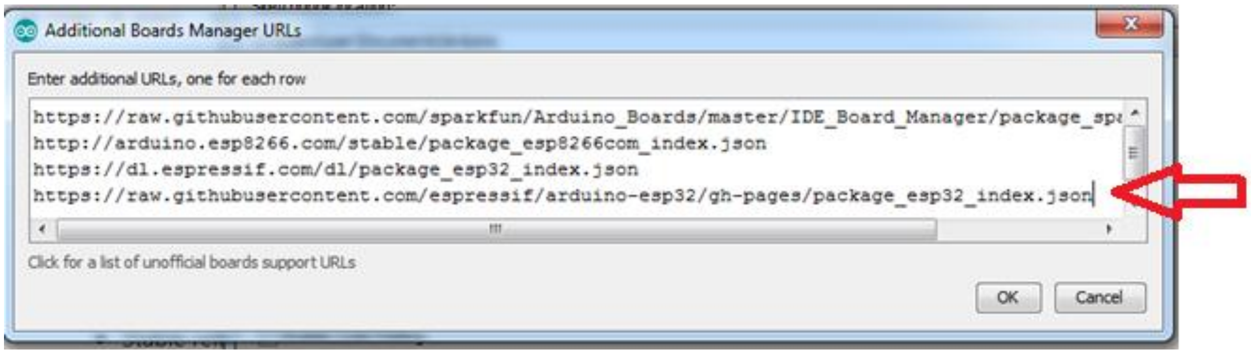
https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json



Additional Boards Manager URLs: https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json

If a text is already in the box (you’ve added some board before), click the icon beside it

and enter the esp32 link above in the next line. Click OK.



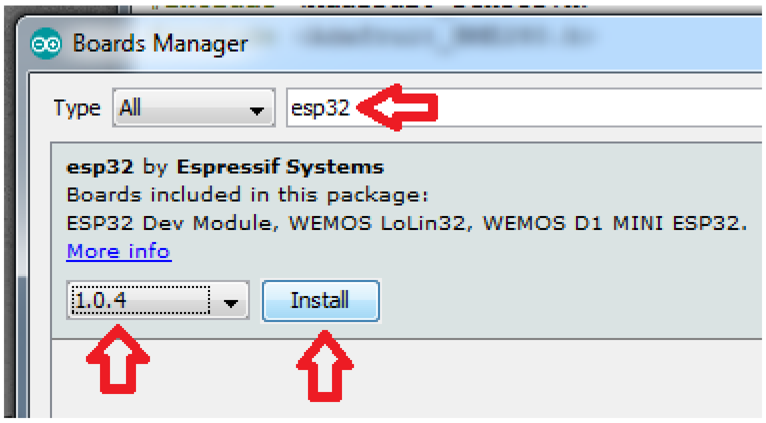
Additional Boards Manager URLs

Enter additional URLs, one for each row

```
https://raw.githubusercontent.com/sparkfun/Arduino_Boards/master/IDE_Board_Manager/package_spa
http://arduino.esp8266.com/stable/package_esp8266com_index.json
https://dl.espressif.com/dl/package_esp32_index.json
https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json
```

Now in Arduino IDE go to **Tools>Board>Boards Manager**.

In the blank window type **esp32**. Click drop-down button choose the latest version and click **Install**. Wait for the installation to finish. You may need to restart your Arduino IDE to take effect off all the changes.


Boards Manager

Type All esp32

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

1.0.4 Install

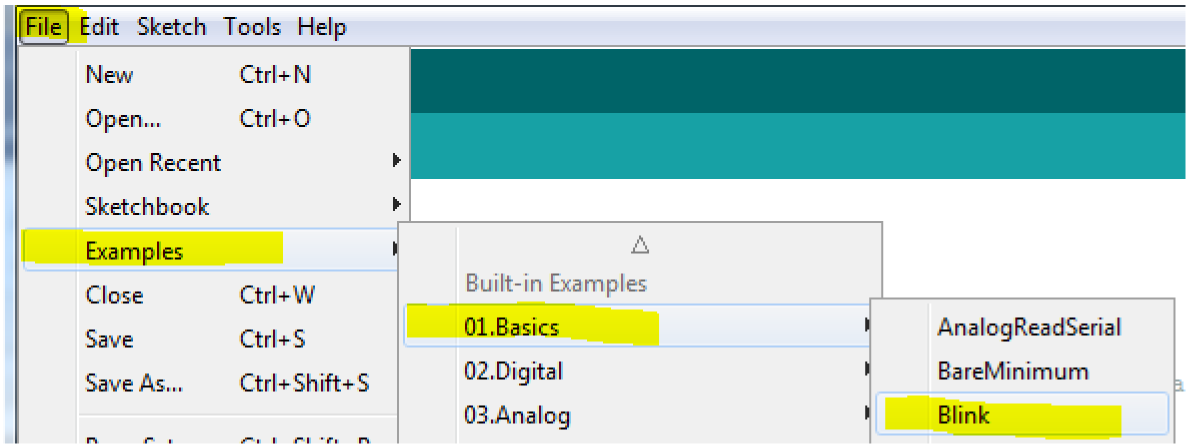
Your ESP32 dev board should now be recognized in the Arduino IDE.

Section 2: Running with Arduino

Here we will run examples to test our ESP32 board using the Arduino IDE.

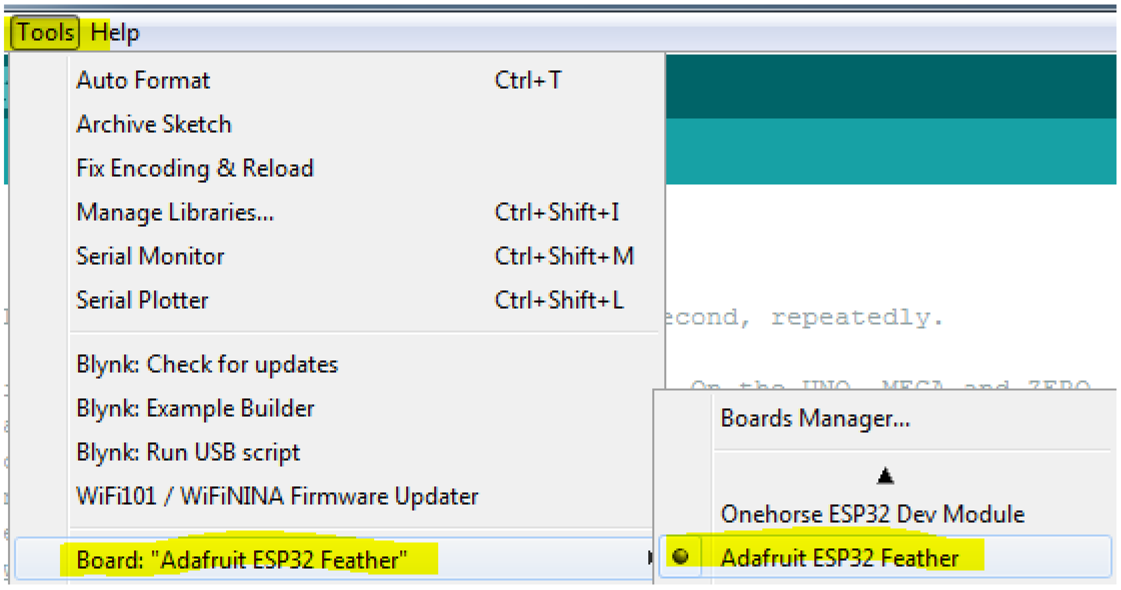
Example 2.1 ESP32 Blink

In your Arduino IDE go to **File>Examples>Basics>Blink**.



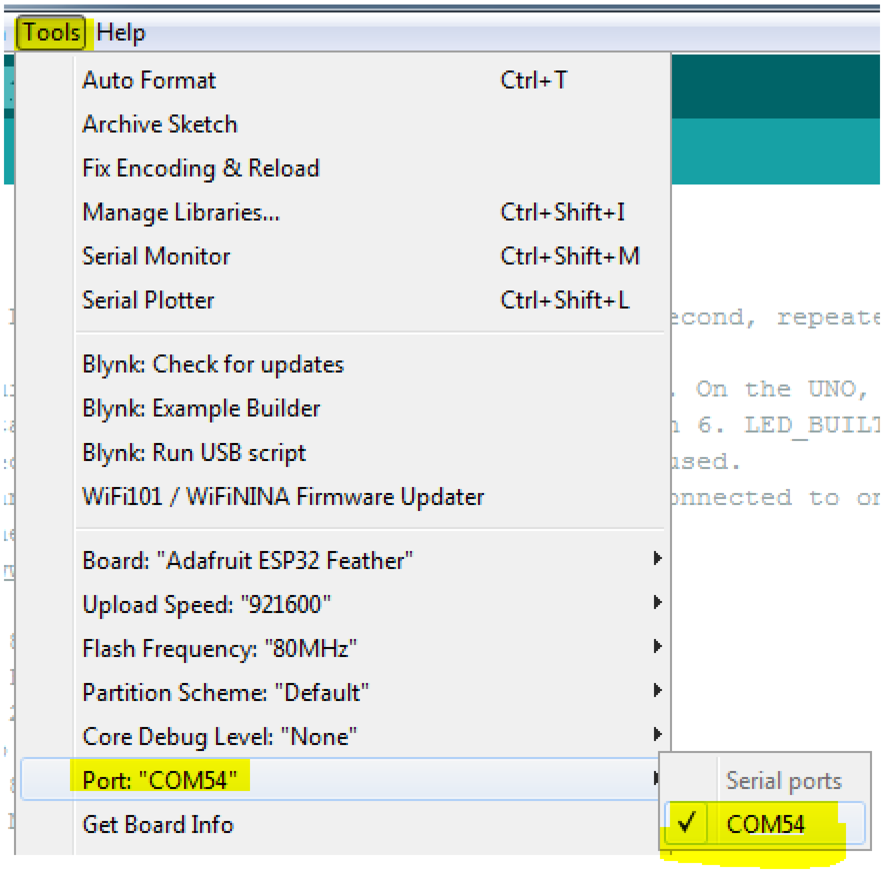
Once open you will see the Blink code!

Now go to **Tools>Board>Adafruit ESP32 Feather**.



Board: "Adafruit ESP32 Feather"

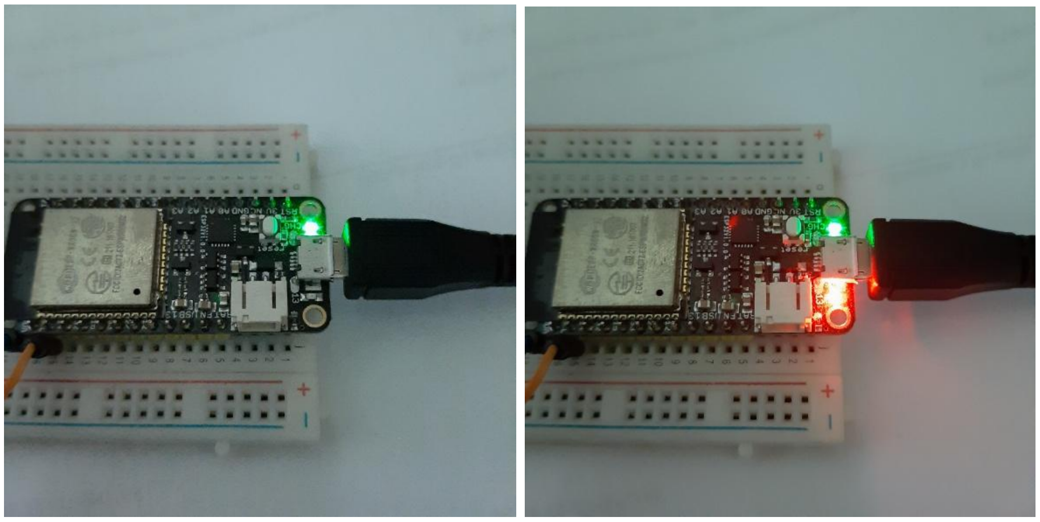
Then, again go **Tools>Port>COMXX ---> (choose the COM port that appeared in your Windows device manager earlier).**



Board: "Adafruit ESP32 Feather"
Upload Speed: "921600"
Flash Frequency: "80MHz"
Partition Scheme: "Default"
Core Debug Level: "None"
Port: "COM54"

Now click **Verify** and **Upload**.

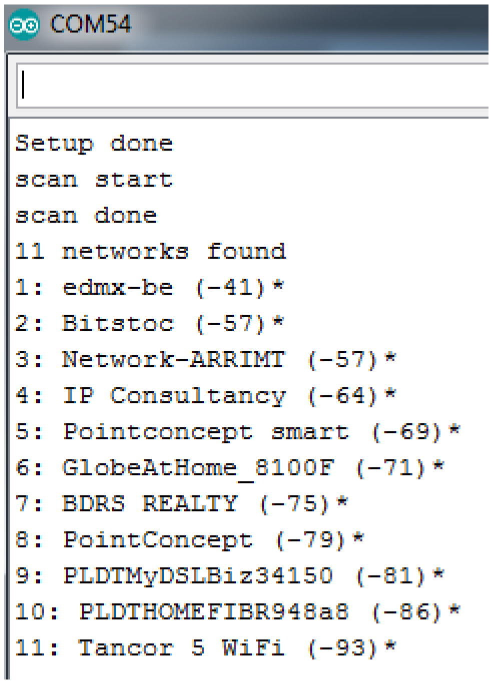
You should see the built-in **LED** in the ESP32 Development Board **blink**!



Example 2.2 ESP32 WiFi Scan

Now let’s try the WiFi capability of the ESP32. In your Arduino IDE go to **File>Examples>WiFi>WiFiScan**.

Click **Verify** and **Upload** to your ESP32 board. You should see the nearby WiFi connections in your area.



```
Setup done
scan start
scan done
11 networks found
1: edmx-be (-41)*
2: Bitstoc (-57)*
3: Network-ARRIMT (-57)*
4: IP Consultancy (-64)*
5: Pointconcept smart (-69)*
6: GlobeAtHome_8100F (-71)*
7: BDRS REALTY (-75)*
8: PointConcept (-79)*
9: PLDTMyDSLBiz34150 (-81)*
10: PLDTHOMEFIBR948a8 (-86)*
11: Tancor 5 WiFi (-93)*
```

Section 3: WiFi and Internet communication example

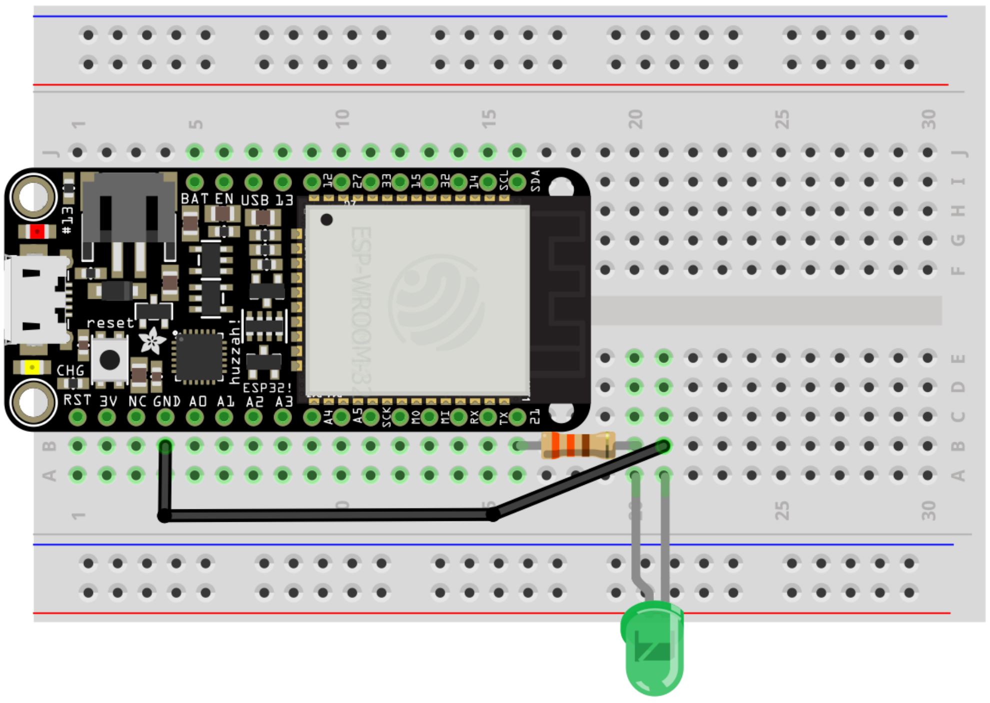
Let's try the WiFi Webserver and Internet connection capability of the ESP32. Here we will host a local webserver (webpage) in our ESP32 chip and we will also ping a live website and get the html content of this live website.

3.1 Example: WiFi local Webserver

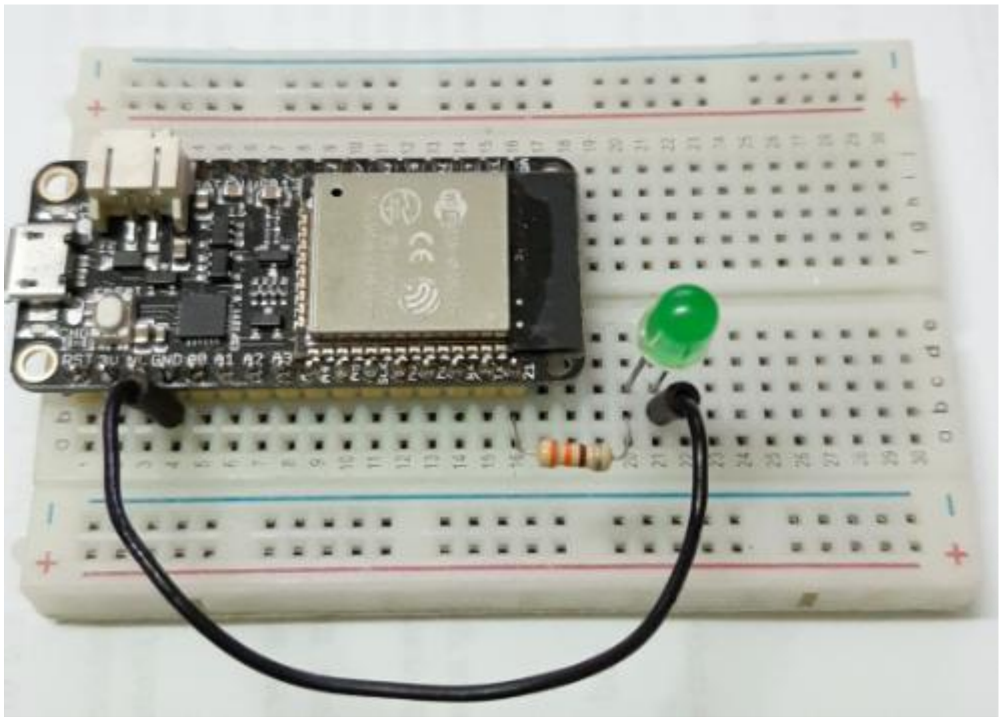
In this example we will host a simple website html page in the ESP32 and have a webpage button in the web browser to turn On and Off an LED on the ESP32 circuit.

ESP32 Circuit

Wire your circuit as follows.



You circuit will look like this.



Here we connect an LED circuit in the **GPIO21** by connecting a 330ohm resistor to GPIO21. The other end of the resistor goes to the positive of the LED and then black wire to ground the LED.

Arduino ESP32 Code

Open a new Arduino window and type the code below but DO NOT UPLOAD IT YET, we still have to edit some information on the code.

```
/*
 WiFi Web Server LED Blink

 A simple web server that lets you blink an LED via the web.
 This sketch will print the IP address of your WiFi Shield (once connected)
 to the Serial monitor. From there, you can open that address in a web browser
 to turn on and off the LED on pin 21.

 If the IP address of your shield is yourAddress:
 http://yourAddress/H turns the LED on
 http://yourAddress/L turns it off

 This example is written for a network using WPA encryption. For
 WEP or WPA, change the Wifi.begin() call accordingly.

 Circuit:
 LED attached to pin 21

 created for arduino 25 Nov 2012
 by Tom Igoe
*/

#include <WiFi.h>

const char* ssid     = "YOUR_WIFI_NAME_HERE";
const char* password = "YOUR_WIFI_PASSWORD_HERE";

WiFiServer server(80);

void setup()
{
 Serial.begin(115200);
 pinMode(21, OUTPUT);      // set the LED pin mode

 delay(10);

 // We start by connecting to a WiFi network

 Serial.println();
 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());

 server.begin();

}
```

(code continuation).

```
int value = 0;

void loop() {
  WiFiClient client = server.available();   // listen for incoming clients

  if (client) {                             // if you get a client,
    Serial.println("New Client.");           // print a message out the serial port
    String currentLine = "";                // make a String to hold incoming data from the client
    while (client.connected()) {            // loop while the client's connected
      if (client.available()) {             // if there's bytes to read from the client,
        char c = client.read();             // read a byte, then
        Serial.write(c);                    // print it out the serial monitor
        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) {
            // HTTP headers always start with a response code (e.g. HTTP/1.1 200 OK)
            // and a content-type so the client knows what's coming, then a blank line:
            client.println("HTTP/1.1 200 OK");
            client.println("Content-type:text/html");
            client.println();

            // the content of the HTTP response follows the header:
            client.print("Click <a href=\"/H\">here</a> to turn the LED on pin 21 on.<br>");
            client.print("Click <a href=\"/L\">here</a> to turn the LED on pin 21 off.<br>");

            // The HTTP response ends with another blank line:
            client.println();
            // break out of the while loop:
            break;
          } 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
        }

        // Check to see if the client request was "GET /H" or "GET /L":
        if (currentLine.endsWith("GET /H")) {
          digitalWrite(21, HIGH);               // GET /H turns the LED on
        }
        if (currentLine.endsWith("GET /L")) {
          digitalWrite(21, LOW);                // GET /L turns the LED off
        }
      }
    }
    // close the connection:
    client.stop();
    Serial.println("Client Disconnected.");
  }
}
```

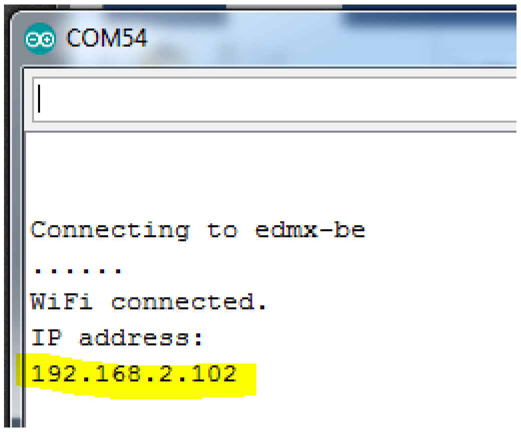
After doing the code above, find the following below from the code:

const char* ssid = "YOUR_WIFI_NAME_HERE";
const char* password = "YOUR_WIFI_PASSWORD_HERE";

change the "YOUR_WIFI_NAME_HERE"; and "YOUR_WIFI_PASSWORD_HERE"; to your own **wifi network name** and **wifi network password** credentials.

Now **Verify** and **Upload** the code to your board.

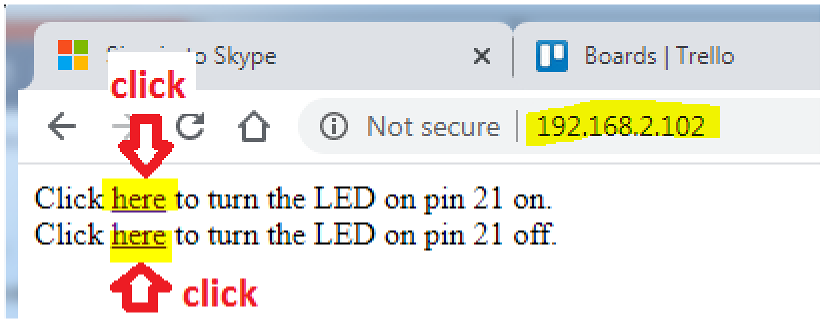
Open your Arduino Serial Monitor and set to **Baud 115200**. Note the **IP address** given.



```
Connecting to edmx-be
......
WiFi connected.
IP address:
192.168.2.102
```

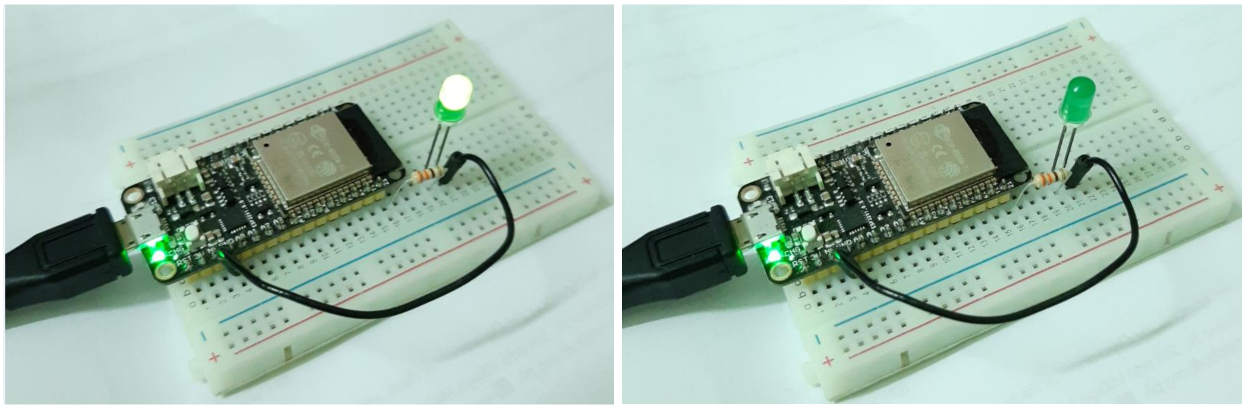
Now open a browser in your computer, laptop or mobile phone that's connected to the same router/wifi network where your esp32 is connected to (from the wifi credentials you inputted earlier).

You should see something like below. Click both the **"here"** button to turn On and Off the LED on your circuit!

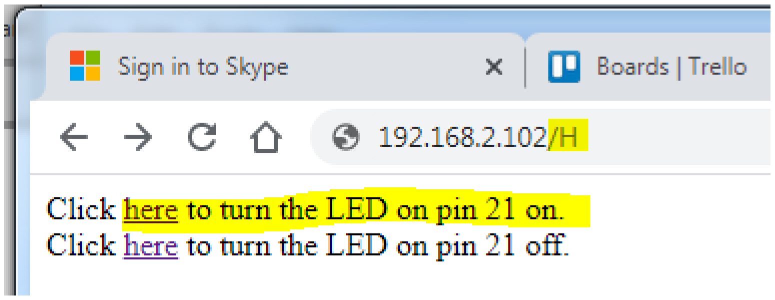


Click here to turn the LED on pin 21 on.
Click here to turn the LED on pin 21 off.

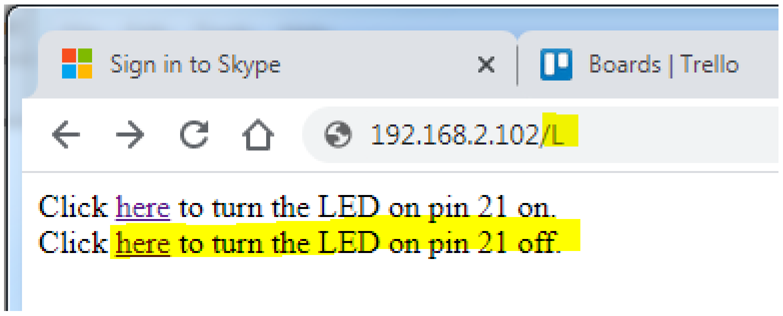
As you press the **"here"** buttons the LED turns on and off!



You will notice in the browser too that to turn on the LED the browser URL sends a combination of the IP address with "/H" and "/L" character to turn on and OFF the LED, this done by the **"here"** buttons. Go and check the HTML part in the Arduino code to learn how it's done.



Click here to turn the LED on pin 21 on.
Click here to turn the LED on pin 21 off.



Click here to turn the LED on pin 21 on.
Click here to turn the LED on pin 21 off.

Try!

Try adding another LED on another pin. How about adding a push button to send data from the circuit to the webserver? Add a sensor too!

Play around on your circuit and code and learn HTML to design your webpage!

3.2 Example: Fetch HTML content of a live website

Now let's get live from the internet! Let's get html content of a live website using our ESP32 board.

Open a new Arduino window and type the code below. You will also need to enter your wifi credentials.

```
#include <WiFi.h>

// WiFi network name and password:
const char * networkName = "YOUR_WIFI_NAME_HERE";
const char * networkPswd = "YOUR_WIFI_PASSWORD_HERE";

// Internet domain to request from:
const char * hostDomain = "example.com";
const int hostPort = 80;

const int BUTTON_PIN = 0;
const int LED_PIN = 13;

void setup()
{
  // Initilize hardware:
  Serial.begin(115200);
  pinMode(BUTTON_PIN, INPUT_PULLUP);
  pinMode(LED_PIN, OUTPUT);

  // Connect to the WiFi network (see function below loop)
  connectToWiFi(networkName, networkPswd);

  digitalWrite(LED_PIN, LOW); // LED off
  Serial.print("Press button 0 to connect to ");
  Serial.println(hostDomain);
}

void loop()
{
  delay(1000); // put a delay every website ping

  digitalWrite(LED_PIN, HIGH); // Turn on LED
    requestURL(hostDomain, hostPort); // Connect to server
  digitalWrite(LED_PIN, LOW); // Turn off LED
}

void connectToWiFi(const char * ssid, const char * pwd)
{
  int ledState = 0;

  printLine();
  Serial.println("Connecting to WiFi network: " + String(ssid));

  WiFi.begin(ssid, pwd);

  while (WiFi.status() != WL_CONNECTED)
  {
    // Blink LED while we're connecting:
    digitalWrite(LED_PIN, ledState);
    ledState = (ledState + 1) % 2; // Flip ledState
    delay(500);
    Serial.print(".");
  }

  Serial.println();
  Serial.println("WiFi connected!");
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());
}
```

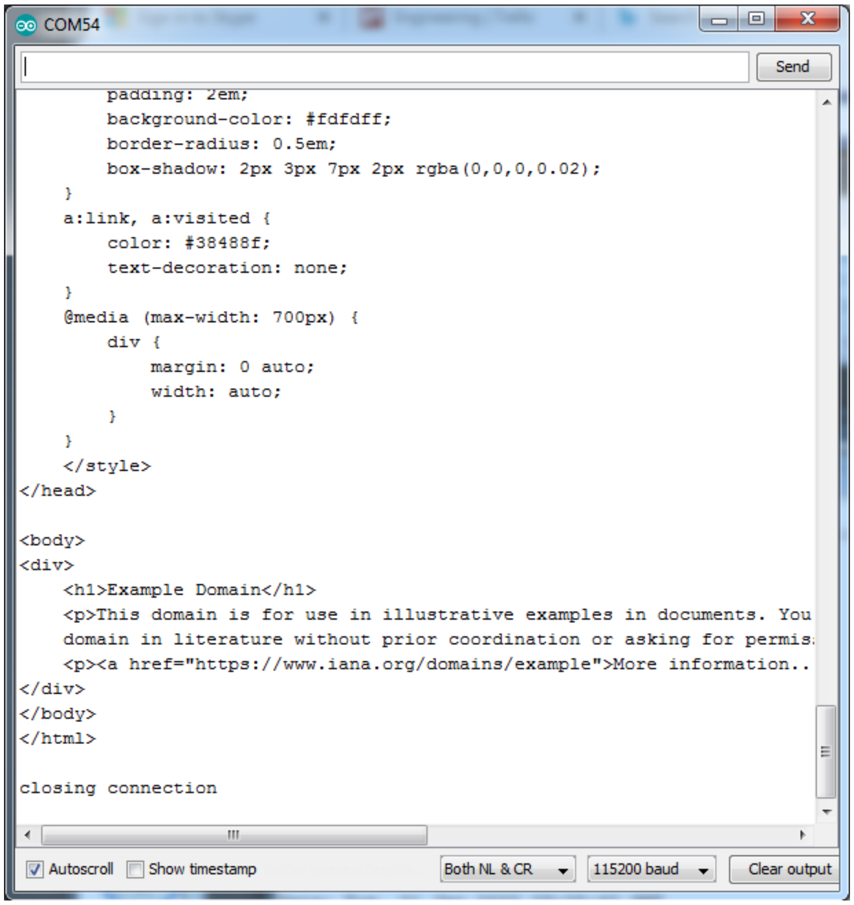
(code continuation)

```
void requestURL(const char * host, uint8_t port)
{
  printLine();
  Serial.println("Connecting to domain: " + String(host));

  // Use WiFiClient class to create TCP connections
  WiFiClient client;
  if (!client.connect(host, port))
  {
    Serial.println("connection failed");
    return;
  }
  Serial.println("Connected!");
  printLine();

  // This will send the request to the server
  client.print((String)"GET / HTTP/1.1\r\n" +
               "Host: " + String(host) + "\r\n" +
               "Connection: close\r\n\r\n");
  unsigned long timeout = millis();
  while (client.available() == 0)
  {
    if (millis() - timeout > 5000)
    {
      Serial.println(">>> Client Timeout !");
      client.stop();
      return;
    }
  }

  // Read all the lines of the reply from server and print them to Serial
  while (client.available())
  {
    String line = client.readStringUntil('\r');
    Serial.print(line);
  }

  Serial.println();
  Serial.println("closing connection");
  client.stop();
}

void printLine()
{
  Serial.println();
  for (int i=0; i<30; i++)
    Serial.print("-");
  Serial.println();
}
```

Before uploading the code, find in the code and change the **"YOUR_WIFI_NAME_HERE"**; and **"YOUR_WIFI_PASSWORD_HERE"**; to your own **wifi network name** and **wifi network password** credentials.

Now **Verify** and **Upload** the code to your board.

Open your Arduino Serial Monitor and set to **Baud 115200**. You should see you are connected to WiFi network and the HTML content of the website **www.example.com**.



COM54

Send

```
        padding: 2em;
        background-color: #fdfdff;
        border-radius: 0.5em;
        box-shadow: 2px 3px 7px 2px rgba(0,0,0,0.02);
    }
    a:link, a:visited {
        color: #38488f;
        text-decoration: none;
    }
    @media (max-width: 700px) {
        div {
            margin: 0 auto;
            width: auto;
        }
    }
    </style>
</head>

<body>
<div>
    <h1>Example Domain</h1>
    <p>This domain is for use in illustrative examples in documents. You
    domain in literature without prior coordination or asking for permis
    <p><a href="https://www.iana.org/domains/example">More information..
</div>
</body>
</html>

closing connection
```

[✓] Autoscroll [ ] Show timestamp — Both NL & CR — 115200 baud — Clear output

Going Further

You can do a lot from this board. This is just a test to try your ESP32.

Go and try the BLE (Bluetooth Low Energy) capabilities of the ESP32 chip. BLE is a great Bluetooth communication technology which increase the range and very low operating power compared to previous Bluetooth version. BLE application can run on small batteries for years!

Go and checkout these BLE examples:

BLE with ESP32 and Arduino by Sparkfun

ESP32 BLE + Android + Arduino IDE by arduinofanboy

ESP32 Bluetooth Low Energy (BLE) on Arduino IDE by randomnerdtutorials

BLE concepts and example with ESP32 by electronics-lab

Other Examples

Another great examples to try with ESP32 Board:

ESP32 Weather Station with BME280 by lastminuteengineers

Weather Station Arduino ESP32 by randomnerdtutorials