ESP32 WIFI/BLE Board v0.9

From Elecrow

Contents

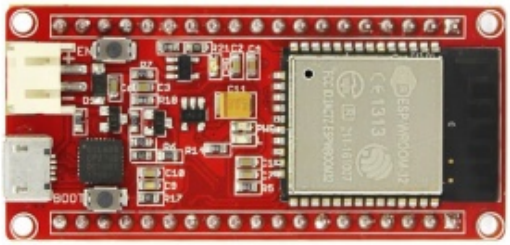
Introduction

ESP-WROOM-32 is a powerful, generic Wi-Fi+BT+BLE MCU module that targets a wide variety of applications, ranging from low-power sensor networks to the most demanding tasks, such as voice encoding, music streaming and MP3 decoding.

The development board breaks out all the module's pins to 0.1" headers and provides a CP2102 USB-TTL serial adapter, programming and reset buttons, and a power regulator to supply the ESP-WROOM-32 with a stable 3.3 V. Espressif doubled-down on the CPU resources for ESP-WROOM-32 with a dual core, running at 160MHz and tons more pins and peripherals.

The integration of Bluetooth, Bluetooth LE and Wi-Fi ensures that a wide range of applications can be targeted, and that the module is future proof: using Wi-Fi allows a large physical range and direct connection to the internet through a Wi-Fi router, while using Bluetooth allows the user to conveniently connect to the phone or broadcast low energy beacons for its detection. The sleep current of the ESP32 chip is less than 5 μA, making it suitable for battery powered and wearable electronics applications. ESP-WROOM-32 supports data rates of up to 150 Mbps, and 22 dBm output power at the PA to ensure the widest physical range. As such the chip does offer industry-leading specifications and the best performance for electronic integration, range, power consumption, and connectivity.

Model: ARS01119B (https://www.elecrow.com/catalog/product/view/id/2392/)



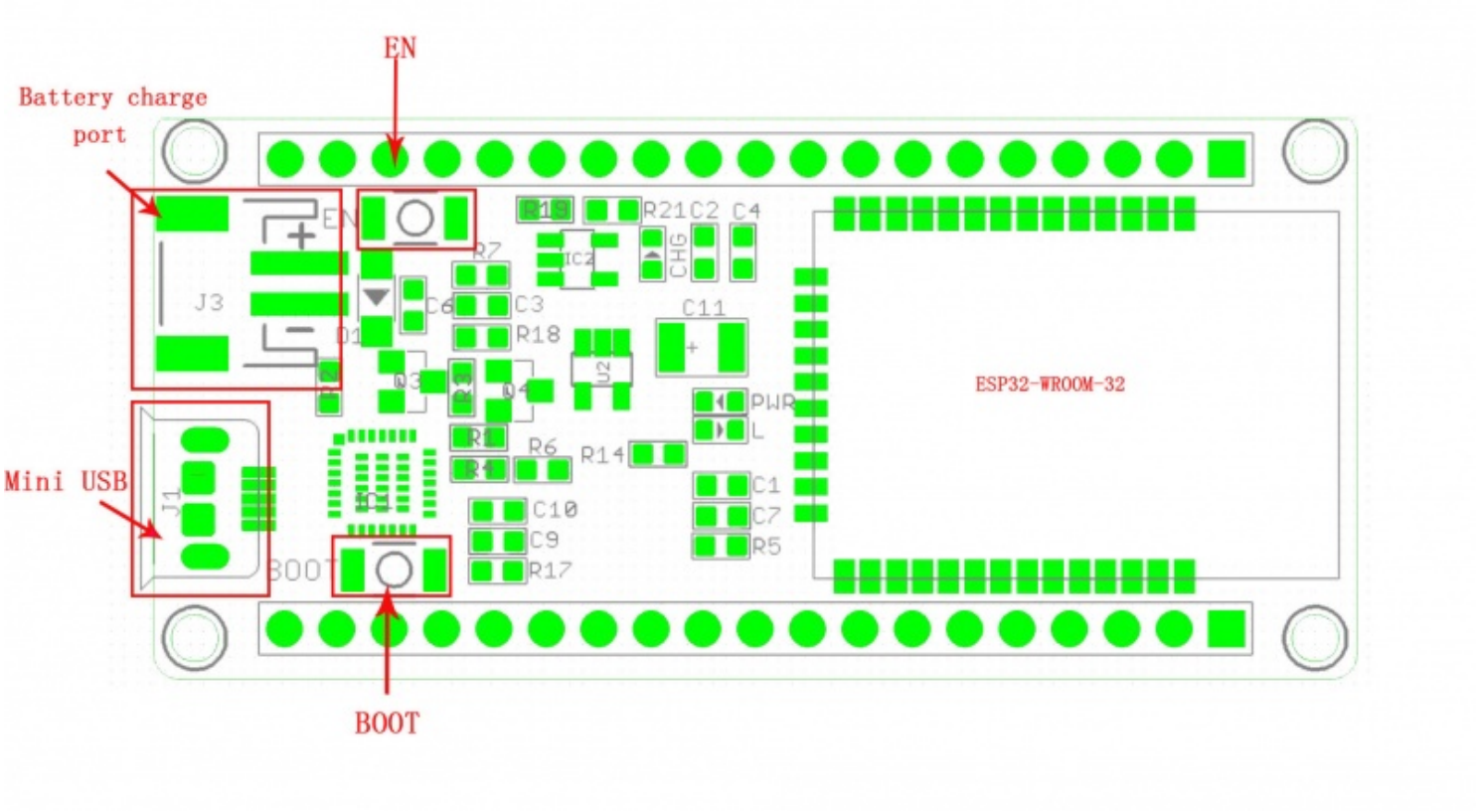
Feactures

- Support SD card, UART, SPI, SDIO, I2C, LED PWM, Motor PWM,I2S, I2C, IRUSB-UART
- Integrated 802.11 BGN WiFi transceiver and dual-mode Bluetooth (classic and BLE).
- Hardware accelerated encryption (AES, SHA2, ECC, RSA-4096)
- Integrated LiPo Battery Charger.
- 10-electrode capacitive touch support
- Up to 240MHz clock frequency and 520kB internal SRAM
- Support Station/SoftAP/SoftAP+Station/P2P
- Support WPA/WPA2/WPA2-Enterprise/WPS
- Support AES/RSA/ECC/SHA
- Include bridge, reset- and boot-mode buttons, LDO regulator and a micro-USB connector.

Specification

- FCC/CE/IC/TELEC/KCC/SRRC/NCC
- Wi-Fi : 802.11 b/g/n/e/i (802.11n up to 150 Mbps)
- A- MPDU and A-MSDU aggregation and 0.4 _s guard interval support
- Frequency range : 2.4 ~ 2.5 GHz
- Bluetooth : v4.2 BR/EDR and BLE specification
- Radio : NZIF receiver with -98 dBm sensitivity, Class-1, class-2 and class-3 transmitter
- Audio : CVSD and SBC
- On-board clock : 40 MHz crystal
- Operating voltage : 2.2 ~ 3.6V
- Operating current : Average: 80 mA
- Operating temperature range : -40°C ~ 85°C

Interface Function



EN Reset button. Pressing this button resets the system.

BOOT Download button. Holding down the Boot button and pressing the EN button initiates the firmware download mode. Then users can download firmware through the serial port

USB USB interface. It functions as the power supply for the board and the communication interface between PC and ESP-WROOM-32.

I/O Most of the pins on the ESP-WROOM-32 are led out to the pin headers on the board. Users can program ESP32 to enable multiple functions such as PWM, ADC, DAC, I2C, I2S, SPI, etc.

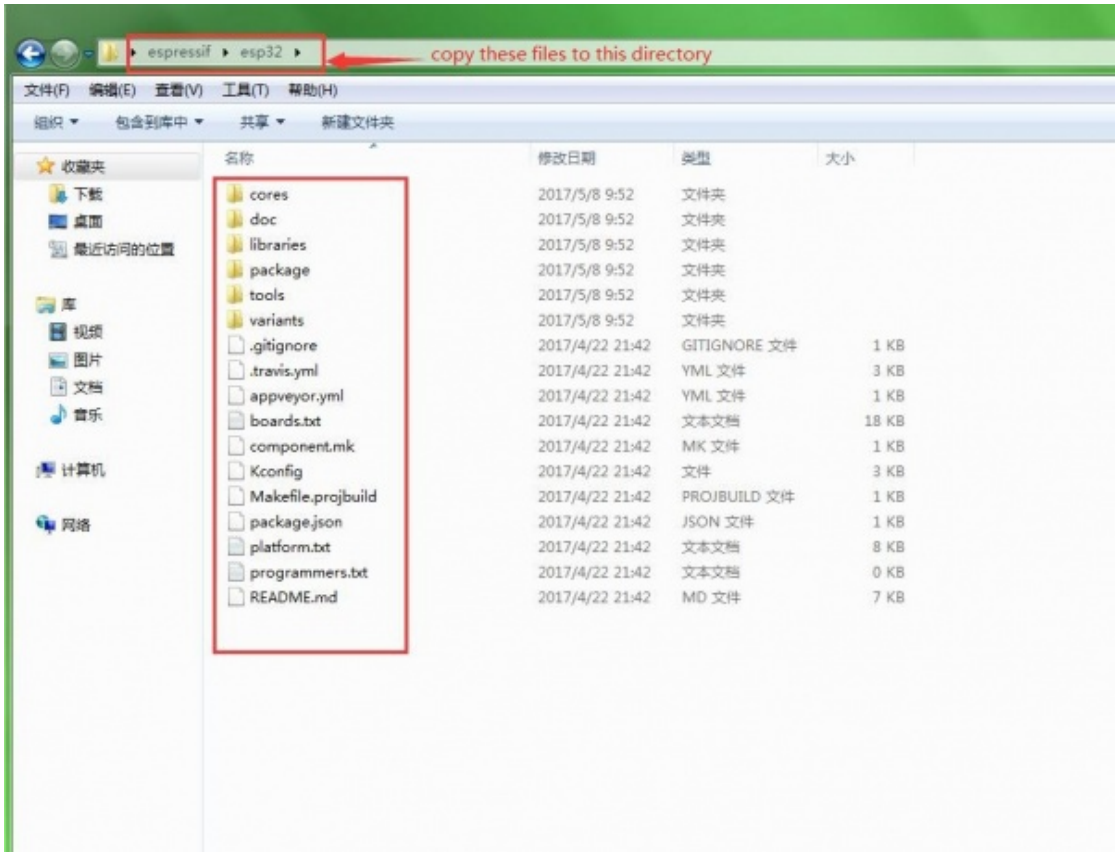
Usage

Installing the ESP32 Arduino Core

This tutorial covers setting up the ESP32 with Arduino IDE, and documents a few simple example sketches to help make your WiFi/BLE microcontroller work.

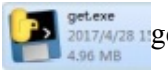
Download the Core

Espressif's official ESP32 Arduino core is hosted here. To install the ESP32 board definitions, you'll need download the contents of the esp32-arduino repository, and place them in a "hardware/espressif/esp32" directory in your Arduino sketchbook directory. First, you need to Create new folder espreesif/esp32, then Download Espressif's official ESP32 Arduino core on here (https://github.com/espressif/arduino-esp32) and unzip it, copy those files to espreesif/esp32 directory.



Copy "espreeif" folder to hardware directory. On windows, that may be C:/program Files/Arduino/hardware , and on Mac that may be /Applications/Arduino.app/Contents/Java/hardware

Install the Xtensa and ESP32 Tools

To compile code for the ESP32, you need the Xtensa GNU compiler collection (GCC) installed on your machine. Windows users can run 

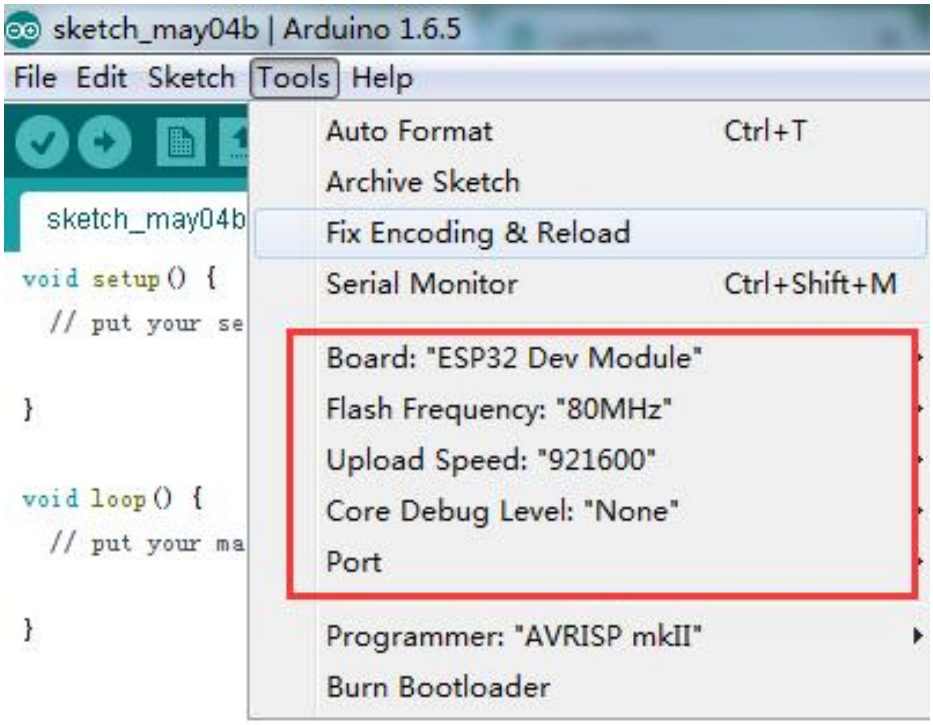
get.exe, found in the "esp32/tools" folder. Mac and Linux users should run the tools/get.py python script to download the tools. Using a terminal, navigate to the esp32/tools folder. Then type:

```
python get.py
```

The "get.py" python script will download the Xtensa GNU tools and the ESP32 software development kit (SDK), and unzip them to the proper location. You should see a few new folders in the "tools" directory, including "dist" and "xtensa-esp32-elf" once it's done.

Example: Blink

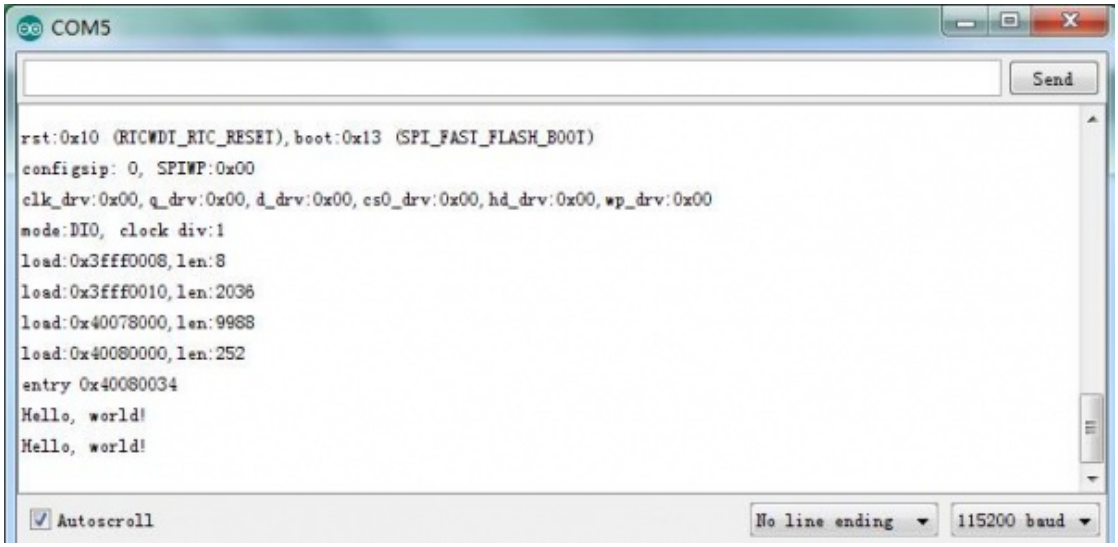
Once the ESP32 Arduino core is installed, you should see an "ESP32 Dev Module" option under your "Tools" > "Board" menu. Select the "Upload Speed"-921600 baud, and serial port.



For this test. Plus, with the ESP32 attached to your computer, it's a good time to test out serial. Copy and paste the example sketch below, into a fresh Arduino sketch.

```
int ledPin = 16;

void setup()
{
    pinMode(ledPin, OUTPUT);
    Serial.begin(115200);
}

void loop()
{
    Serial.println("Hello, world!");
    digitalWrite(ledPin, HIGH);
    delay(500);
    digitalWrite(ledPin, LOW);
    delay(500);
}
```

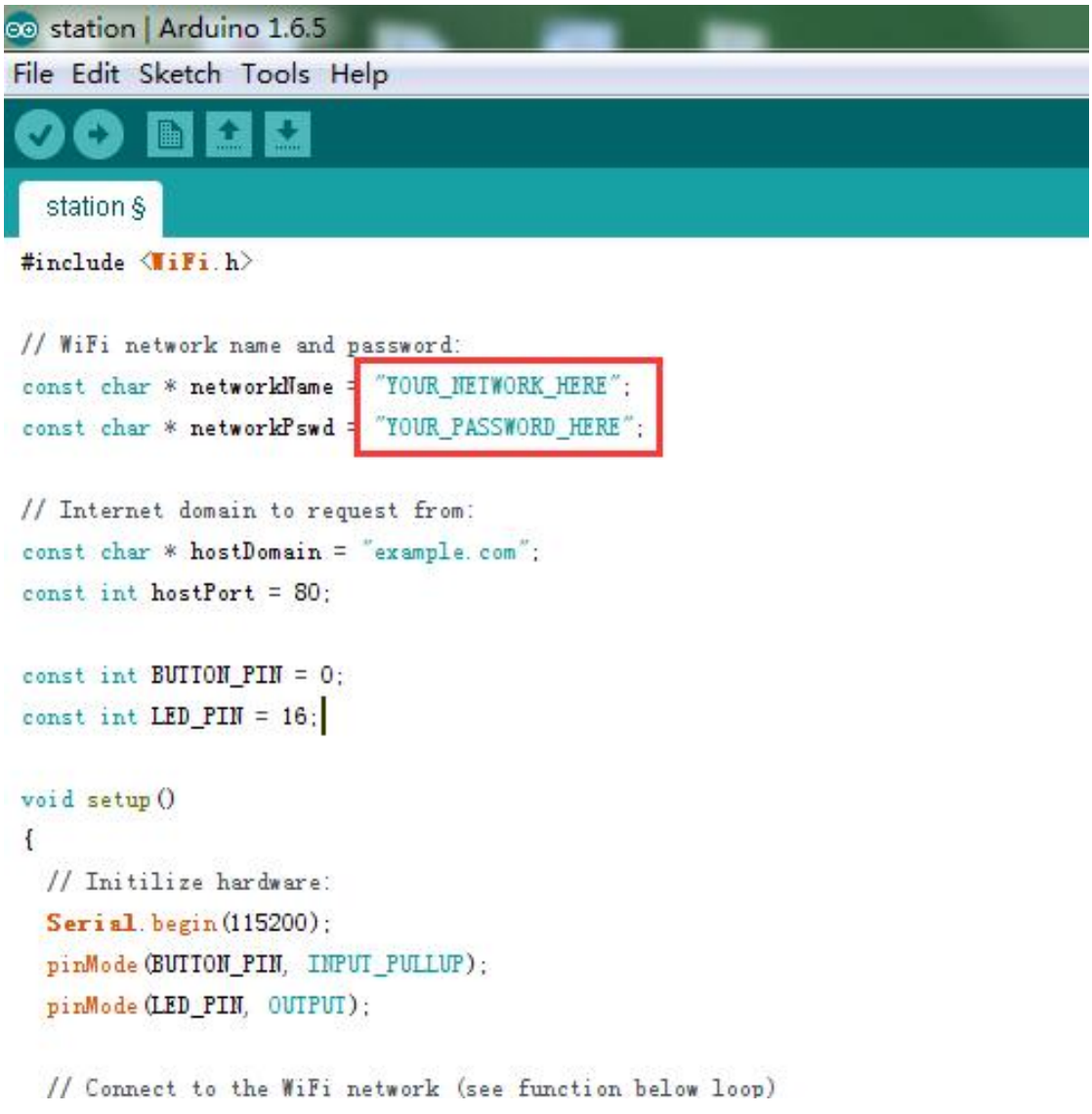
Upload the code! Once the code finishes transferring, open the serial monitor and set the baud rate to 115200. You should see the blue LED "L" keep flashing and "Hello, world" be printed on serial port interface.



TEST: WIFI

Station

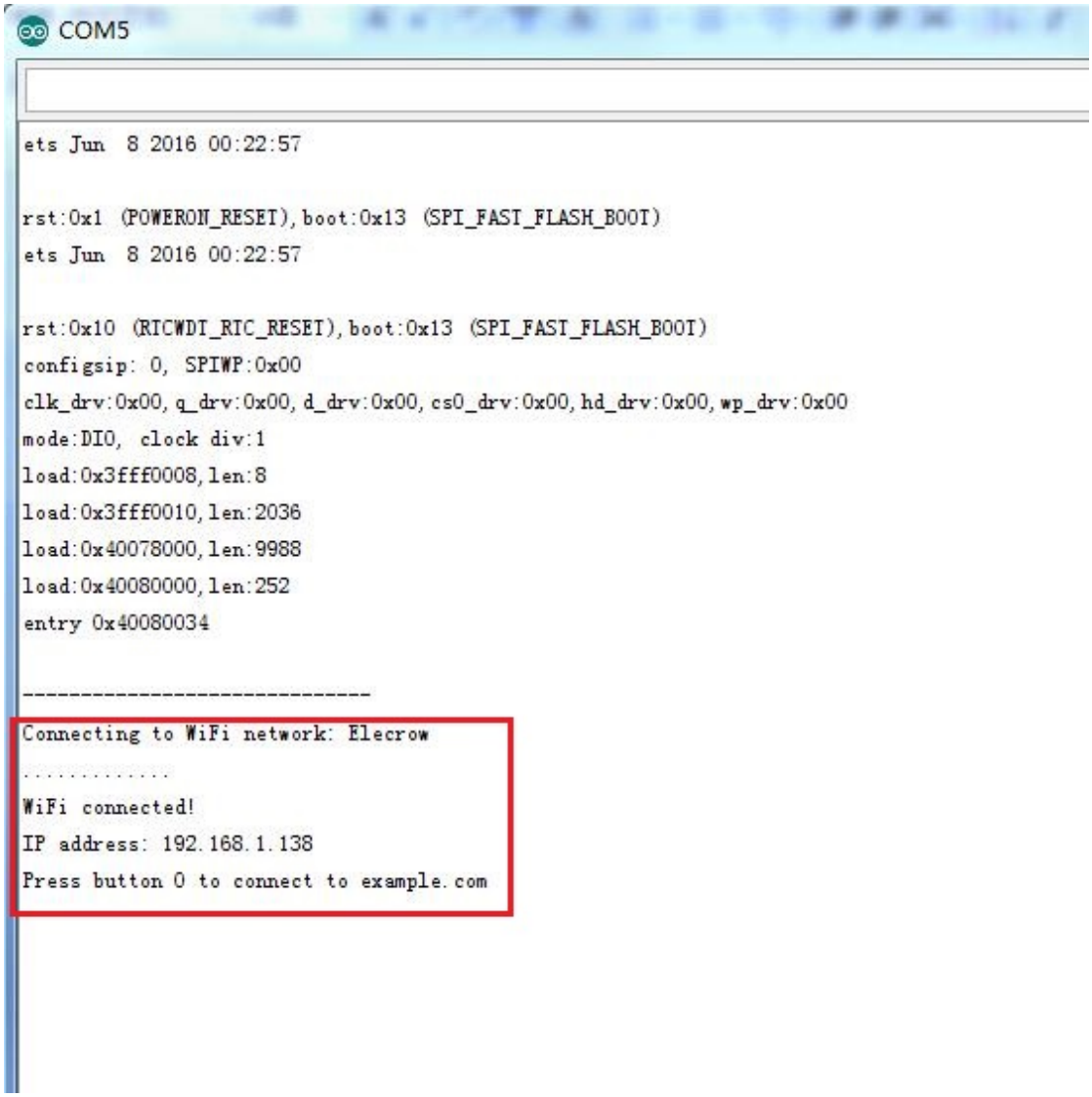
Open the “station.ino” code, you need to change message in the code. Make sure you fill in the YOUR_NETWORK_HERE and YOUR_PASSWORD_HERE variables with the name (SSID) and password of your WiFi network!



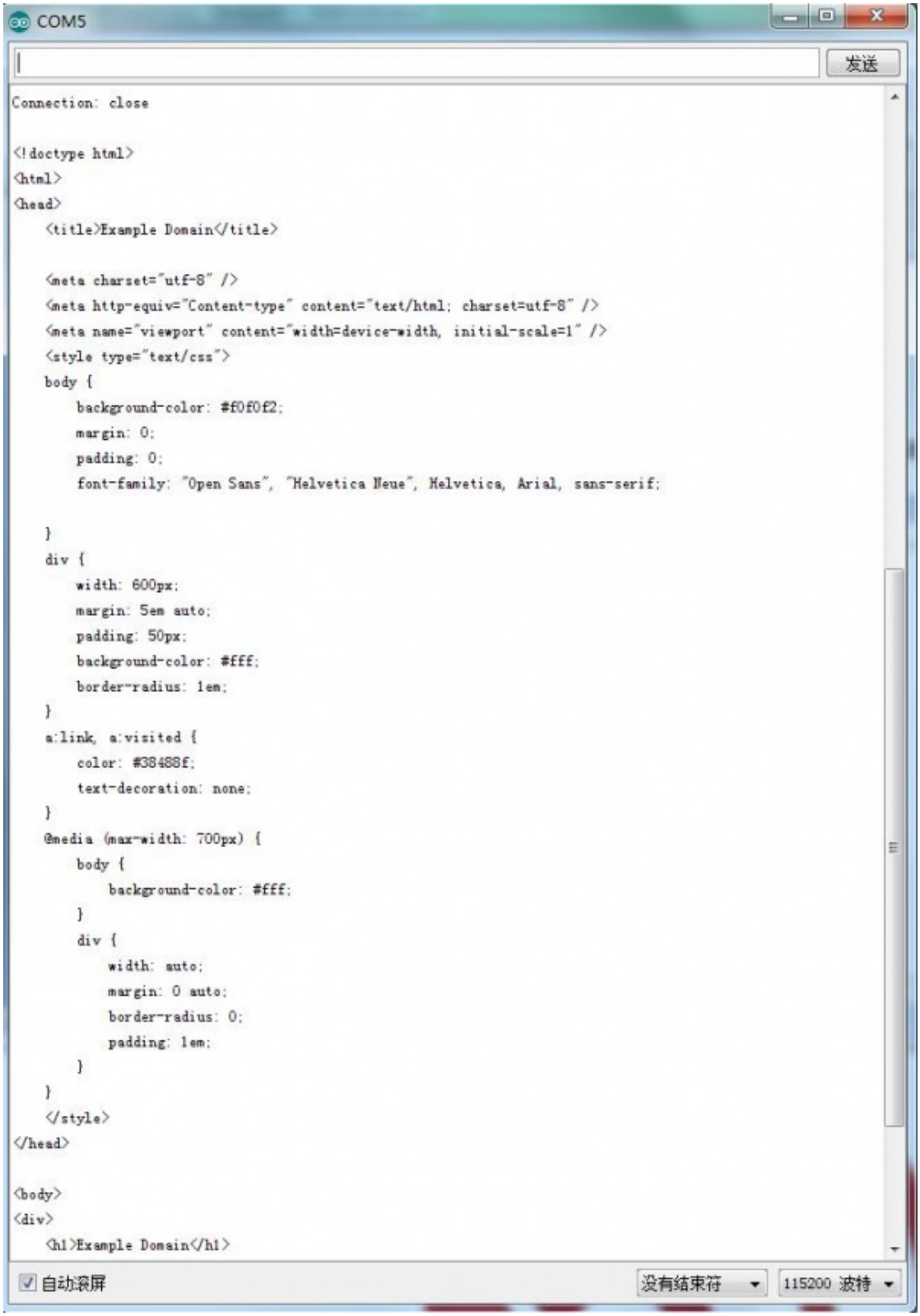
```
#include <WiFi.h>

// WiFi network name and password:
const char * networkName = "YOUR_NETWORK_HERE";
const char * networkPswd = "YOUR_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 = 16;

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()
{
  if (digitalRead(BUTTON_PIN) == LOW)
  { // Check if button has been pressed
    while (digitalRead(BUTTON_PIN) == LOW)
      ; // Wait for button to be released

    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());
}

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();
}
```

Then open the serial monitor . 



When the "WiFi connected!" is printed on the serial monitor ,press the BOOT button, your ESP32 wifi/ble module will send a request to example.com and you can see a few message of HTML is printed the serial monitor.



AP

Open and upload the “ap.ino” code.

```
#include "WiFi.h"

#define AP_SSID  "esp32"

enum {STEP_AP, STEP_END ,STEP};

void onButton(){
```

```
  static uint32_t step = STEP_AP;
  switch(step){

    case STEP_AP://AP Only
      WiFi.mode(WIFI_AP);
      Serial.println("** Starting AP");
      WiFi.softAP(AP_SSID);
    break;
    case STEP_END:
    Serial.println("** Stopping AP");
      WiFi.mode(WIFI_OFF);

    default:
    break;
  }

 if(step == STEP){
    step = STEP_AP;
  } else {
    step++;
  }
  //little debounce
  delay(100);
}

void WiFiEvent(WiFiEvent_t event){
    switch(event) {
        case SYSTEM_EVENT_AP_START:
            Serial.println("AP Started");
            WiFi.softAPsetHostname(AP_SSID);
            break;
        case SYSTEM_EVENT_AP_STOP:
            Serial.println("AP Stopped");
            break;

        default:
            break;
    }
}

void setup() {
    Serial.begin(115200);
    pinMode(0, INPUT_PULLUP);
    WiFi.onEvent(WiFiEvent);
    Serial.print("ESP32 SDK: ");
    Serial.println(ESP.getSdkVersion());
    Serial.println("Press the button to select the next mode");
}

void loop() {
    static uint8_t lastPinState = 1;
    uint8_t pinState = digitalRead(0);
    if(!pinState && lastPinState){
        onButton();
    }
    lastPinState = pinState;
}
```

ap | Arduino 1.6.5

File Edit Sketch Tools Help

ap §

```
// limitations under the License.

#include "WiFi.h"
//#define STA_SSID "Elecrow"
//#define STA_PASS "elecrow2014"
#define AP_SSID  "esp32"

enum {STEP_AP, STEP_END ,STEP};

void onButton(){
  static uint32_t step = STEP_AP;
  switch(step){
  /*  case STEP_BTON://BT Only
      Serial.println("** Starting BT");
      btStart();
    break;
    case STEP_BTOFF://All Off
      Serial.println("** Stopping BT");
      btStop();
```

Open the serial monitor.

Then press the BOOT key to start AP pattern.

```
rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
ets Jun  8 2016 00:22:57

rst:0x10 (RTCWDT_RTC_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0x00
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff0008,len:8
load:0x3fff0010,len:2036
load:0x40078000,len:9988
load:0x40080000,len:252
entry 0x40080034
ESP32 SDK: v2.0-rc1-443-g3cad00f-dirty
Press the button to select the next mode
** Starting AP
AP Started
```

自动滚屏 没有结束符 115200 波特

The “AP started “ will be printed on the serial minotor. Now you can turn on mobile phones to search WIFI “esp32” and connect to it.

Resource

- Code for arduino (https://www.elecrow.com/wiki/index.php?title=File:Code_for_arduino.zip)
- ESP32S_WIFI_BLE_Board_v1.0_Eagle+schematic (https://www.elecrow.com/wiki/index.php?title=File:ESP32S_WIFI_BLE_Board_v1.0.zip)
