LABORATOR VII – PLACA ESP32 și WI-FI

1. Introducere

Scopul acestei lucrări de laborator este de a vă familiariza cu o nouă placă de dezvoltare, bazată pe microcontrolerul ESP32. Acest microcontroller suportă conexiuni wireless precum WiFi, Bluetooth, sau Bluetooth Low Energy (BLE), fiind astfel foarte potrivit pentru aplicații IoT. Microcontrolerul are două nuclee ce operează la 240 MHz, fiecare fiind echipat cu un microprocesor pe 32 de biți de tip Tensilica Xtensa LX6, și poate funcționa în medii industriale cu temperaturi de la -40 la 125 grade Celsius. ESP32 are un consum redus de energie, între 160 mA și 260 mA.

În afara modului activ, microcontrolerul are patru moduri de funcționare cu putere redusă, incluzând modul Deep Sleep, cu un consum de mai puțin de 0.15 mA.

Controlerul ESP32 este de asemenea echipat cu interfețe seriale clasice (cu fire), de tip SPI, I2C și UART.

Placa de dezvoltare trebuie alimentată cu o tensiune între 2.3 și 3.3V, iar pinii acesteia nu acceptă tensiunea de 5 V. De obicei se folosește o sursă de alimentare stabilizată de 3.3 V.

Placa de dezvoltare pe care o vom utiliza în acest laborator este ESP32 Devkit V1, prezentată în Figura 1. Programatorul poate utiliza pinii de uz general (GPIO) pentru intrare și ieșire digitală, sau pentru citirea semnalelor analogice (ADC), sau pentru interfațarea dispozitivelor ce folosesc interfețele seriale SPI (albastru), I2C (roz) sau UART (galben).

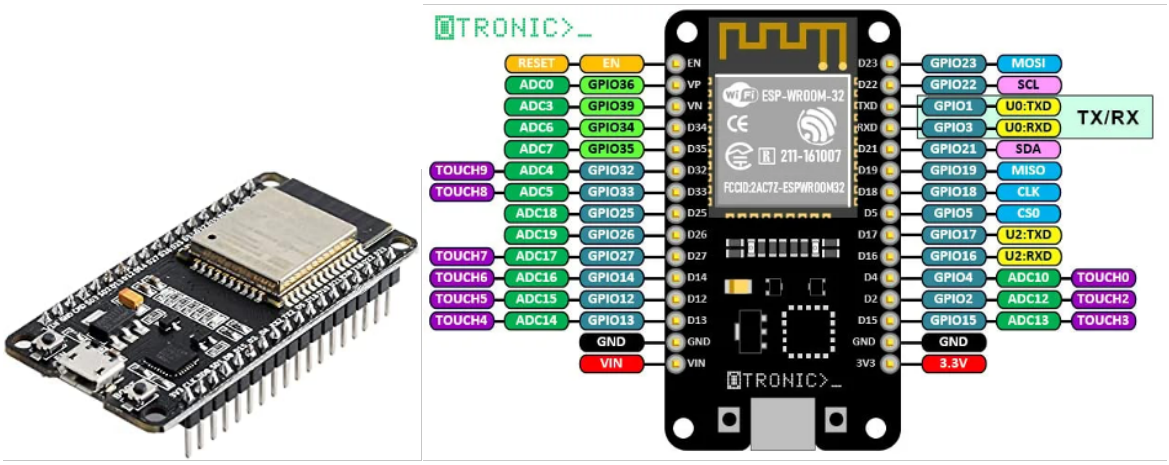


Figure 1. Placa de dezvoltare ESP32 Devkit V1 și pinii acesteia.

2. Montajul fizic pentru activitatea de laborator

Pentru ușurința utilizării și pentru a preveni defectele în timpul activităților de laborator plăcile ESP32 au fost pre-asamblate în structuri ce includ:

- Placa de dezvoltare ESP-WROOM-32 V1
- Un afișor OLED de tip SSD1306, conectat la placa ESP prin interfața 2C
- O sursă de alimentare YwRobot 545043, pentru a putea utiliza alimentatoare externe cu tensiune variabilă
- Un breadboard 30 x (10 + 4)

Montajul este prezentat în Figura 2, iar schema lui este prezentată în Figura 3.

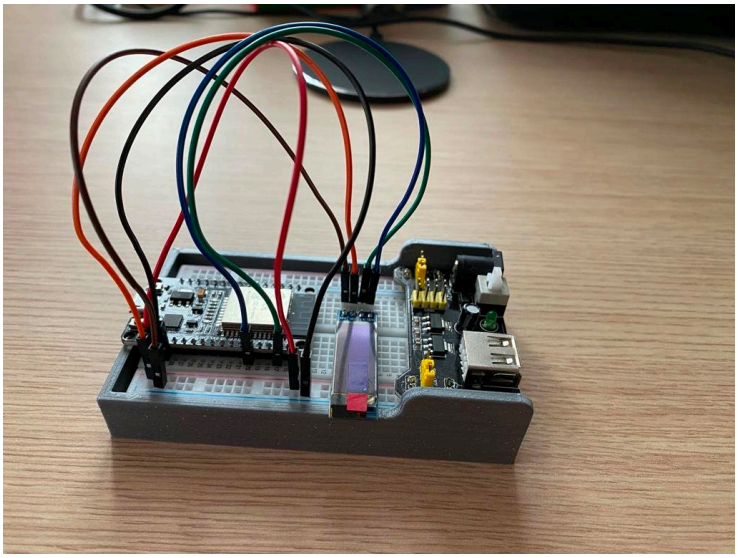


Figura 2. Montajul preasamblat bazat pe ESP 32.

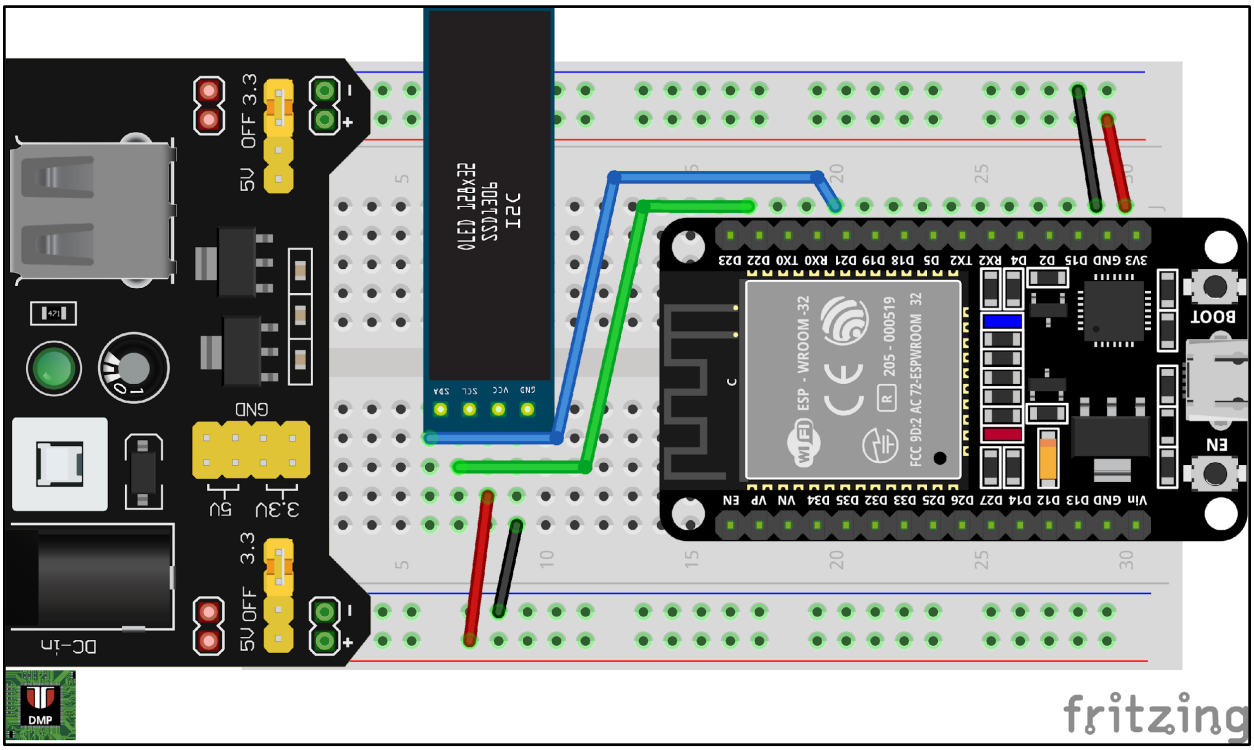


Figura 3. Diagrama montajului preasamblat.

Afișorul

Afișorul SSD1306 este un tip de afișor de tip Organic LED (OLED), ce include semnalele necesare pentru afișare, controlul contrastului, o memorie RAM internă și 256 nivele de intensitate ajustabilă.

Figura 4. Afișorul OLED.

Afișorul OLED este o matrice de puncte ce suportă 128 de coloane și 32 de rânduri pentru a afișa orice formă grafică, și beneficiază de susținerea unei comunități puternice pentru crearea de fonturi sau pentru a transforma imaginile în formate suportate de către afișor.

Afișorul comunică cu ESP32 prin intermediul interfeței I^2C, și poate accepta pe pinul VCC atât tensiunea de 3.3 V cât și cea de 5 V.

Sursa de alimentare

Sursa de alimentare YwRobot 545043 este bazată pe modelul MB V2 și este proiectată pentru a alimenta breadboard-urile.

Această sursă primește curent prin intermediul unui conector rotund (diametru exterior 5.5 mm, diametru interior 2.1 mm), la o tensiune ce poate varia între 6.5 V și 12 V, și produce 3.3 V sau 5 V, curent continuu, cu un amperaj maxim de 700 mA, și o putere maximă de 3.5 W, pe două șine de alimentare.

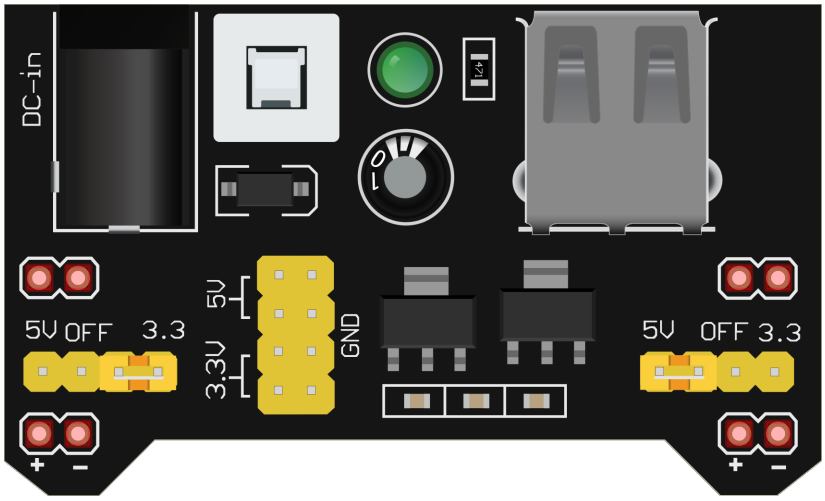


Figura 5. Adaptorul de alimentare pentru breadboard.

Ambele șine pot fi configurate independent prin intermediul unor jumperi (aceștia selectează tensiunea de ieșire pe cele două șine, 3.3 V, 5 V, sau dezactivat. **Vă rugăm nu modificați poziția jumperilor, aceștia sunt poziționați pe 3.3 V, tensiunea potrivită pentru ESP32**). Sursa mai are și un port USB de tip A, pentru alimentarea altor dispozitive. **Acest port este doar pentru a furniza tensiune, nu poate fi utilizat pentru a furniza curent în circuit!**

Sursa de alimentare are un buton de pornire / oprire, și un LED care indică dacă circuitul este alimentat sau nu.

Breadboard

Placa breadboard folosită în acest montaj este numită și breadboard jumătate, pentru că are "doar" 30 de rânduri, și 14 coloane (a – e + f – j + 2 x (+,-)) pentru conectarea pinilor.

Spre deosebire de breadboard-urile pe care le-ați utilizat deja, aceasta are coloane dedicate de tip VCC și GND pe cele două margini lungi, unde pinii sunt conectați în coloane în loc de rânduri. Restul pinilor sunt conectați ca de obicei, orizontal, împărțiți în două jumătăți de rând.

Imaginea de mai jos pune în evidență conexiunile individuale pe verticală - albastru pentru *minus (-)* și roșu pentru *plus (+)* – și conexiuni orizontale – verde pentru coloane a – e și galben pentru coloanele f – j.

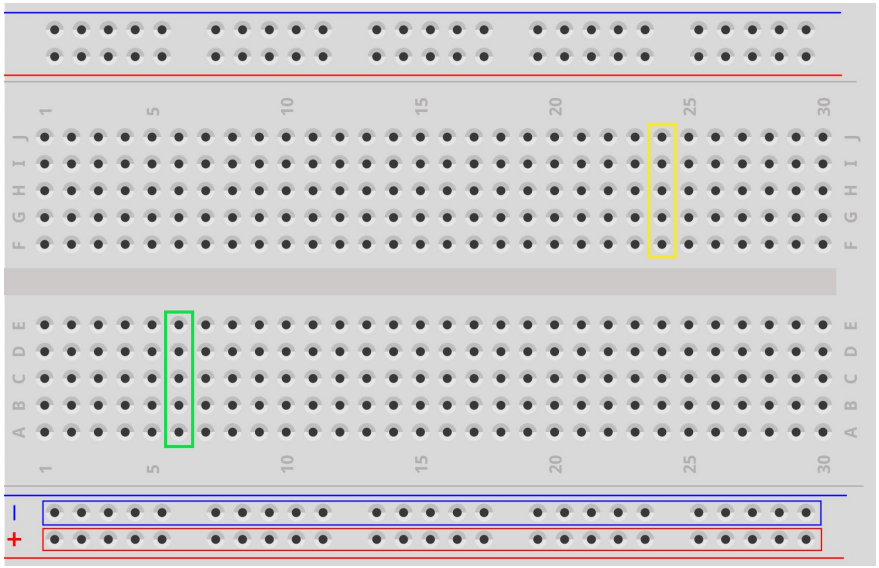


Figura 6. Breadboard cu șine de alimentare.

Separatorul din mijlocul breadboard-ului nu este doar un ajutor vizual, ci este un separator al conexiunilor. De exemplu, un semnal conectat printr-un fir la unul din pinii E va fi conectat și la pinii A…D, dar nu și la pinii F…J, de pe cealaltă parte a breadboard-ului.

3. Configurarea mediului Arduino pentru ESP32

Driver pentru Windows: https://www.silabs.com/documents/public/software/CP210x_Windows_Drivers.zip

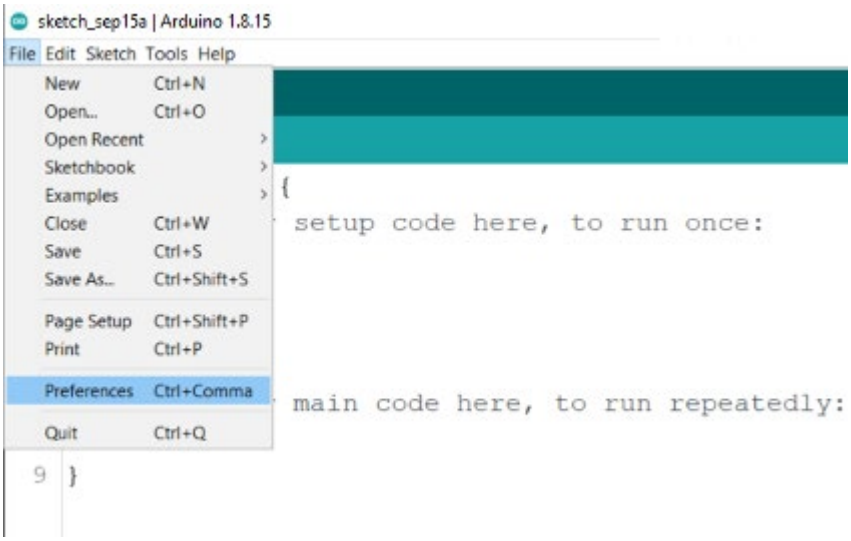
Arduino IDE v1.x

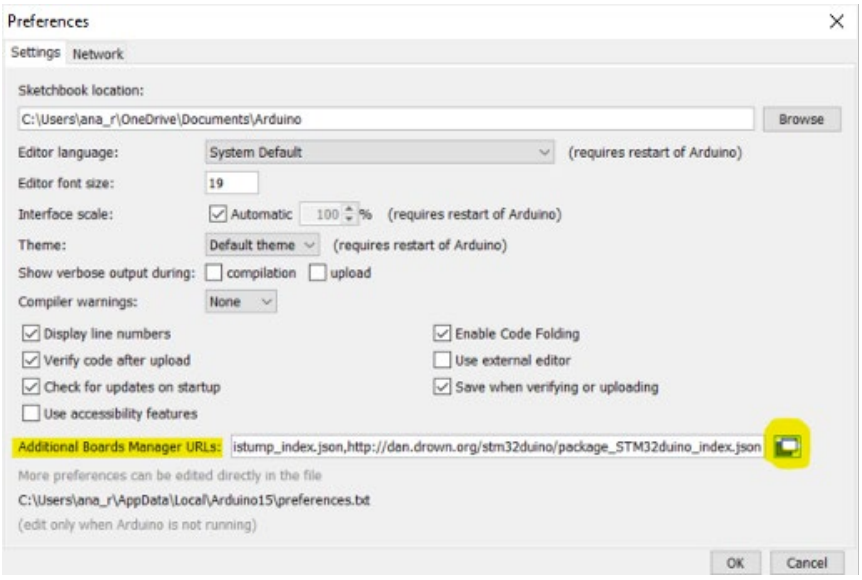
Mediul de programare Arduino IDE este capabil să funcționeze cu noua placă de dezvoltare, dar necesită configurare suplimentară. Pentru instalarea familiei de plăci bazate pe ESP32, dechideți meniul **File / Preferences**, apoi lista **Additional Boards Manager URLs**, și tastați următorul link:

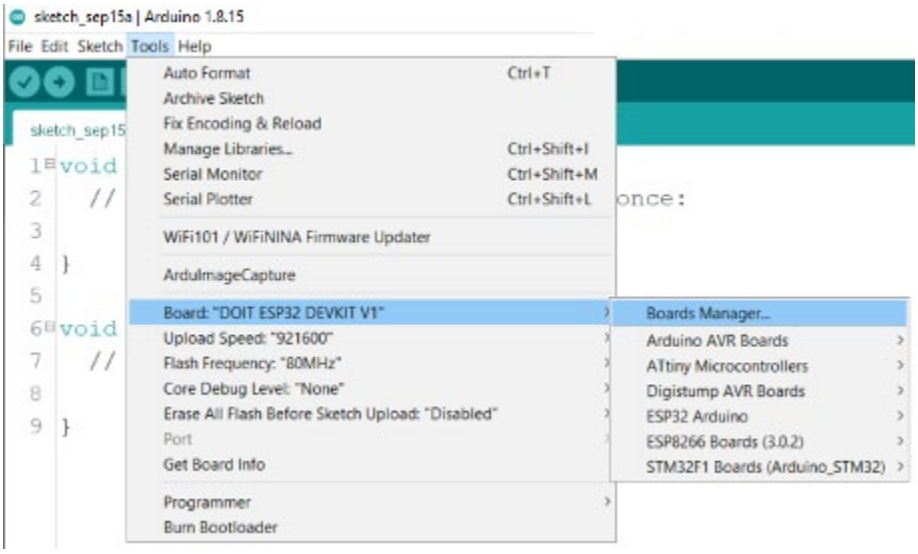
https://dl.espressif.com/dl/package_esp32_index.json

Pasul următor este să deschideți meniul **Tools / Board/Board Manager**, să căutați după familia de plăci ESP32, și să instalați ultima versiune de software.

Pașii de instalare sunt ilustrați în Figura 7.




Preferences
Settings
Network
Sketchbook location:
C:\Users\ana_r\OneDrive\Documents\Arduino
Browse
Editor language:
System Default
(requires restart of Arduino)
Editor font size:
19
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
Enable Code Folding
Verify code after upload
Use external editor
Check for updates on startup
Save when verifying or uploading
Use accessibility features
Additional Boards Manager URLs:
istump_index.json,http://dan.drown.org/stm32duino/package_STM32duino_index.json
More preferences can be edited directly in the file
C:\Users\ana_r\AppData\Local\Arduino15\preferences.txt
(edit only when Arduino is not running)
OK
Cancel


sketch_sep15a | Arduino 1.8.15
File Edit Sketch Tools Help
sketch_sep15
1 void
2 //
3
4 }
5
6 void
7 //
8
9 }
once:
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
ArduImageCapture
Board: "DOIT ESP32 DEVKIT V1"
Upload Speed: "921600"
Flash Frequency: "80MHz"
Core Debug Level: "None"
Erase All Flash Before Sketch Upload: "Disabled"
Port
Get Board Info
Programmer
Burn Bootloader
Boards Manager...
Arduino AVR Boards
ATtiny Microcontrollers
Digistump AVR Boards
ESP32 Arduino
ESP8266 Boards (3.0.2)
STM32F1 Boards (Arduino_STM32)

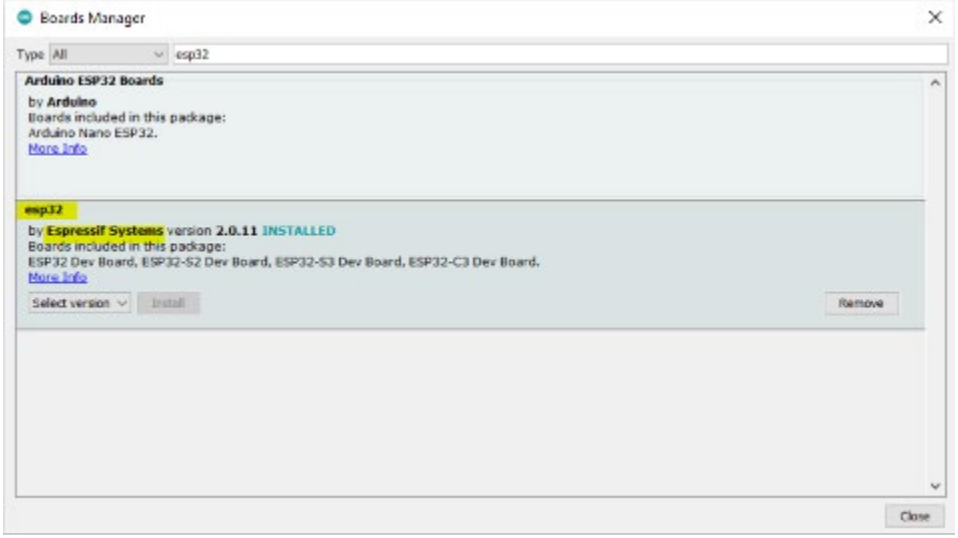


Figura 7. Configurarea mediului Arduino IDE v1.x pentru plăcile ESP32.

După ce ați realizat configurarea, trebuie repornit mediul Arduino IDE, și veți putea folosi noua familie de plăci, disponibilă în meniul

After the installation is complete, you have to restart the Arduino IDE, and a new family of boards will become available in the menu **Tools / Board / ESP 32 Arduino**. Din această listă, alegeți placa **DOIT ESP32 DEVKIT V1**.

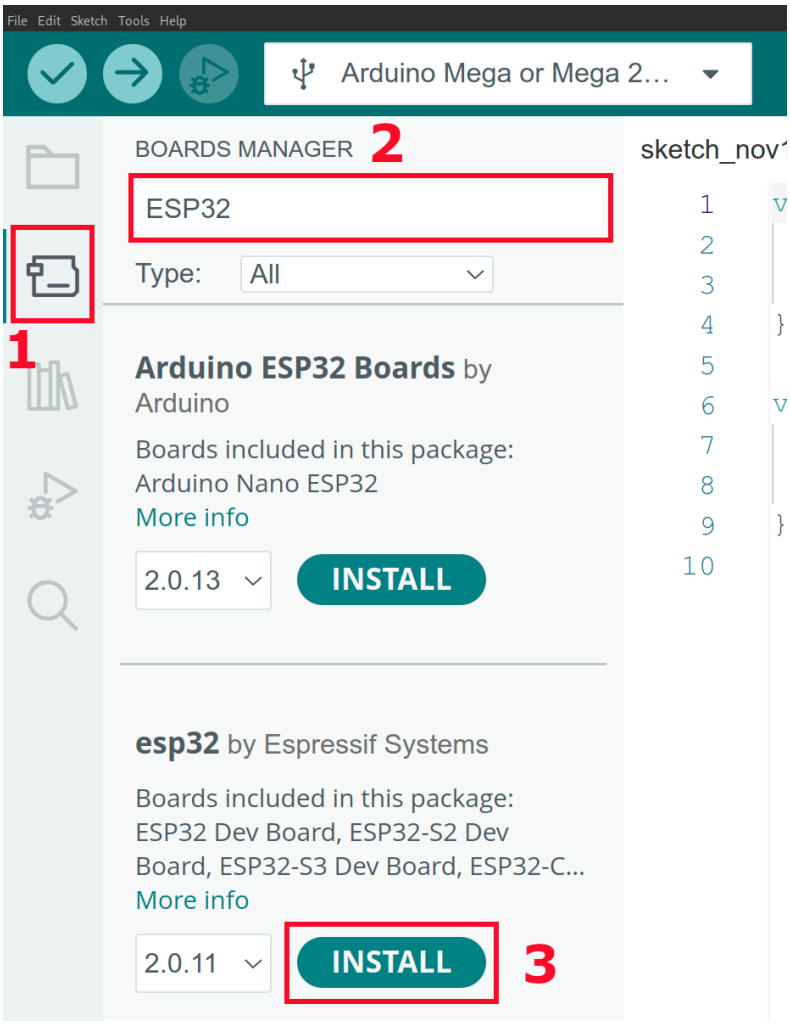
Arduino IDE v2.x

Din meniul vertical, selectați iconul

From the left vertical menu, click on the board manager icon, care va deschide un panou nou. Tastați în căsuța de căutare "ESP32", apoi instalați biblioteca "**esp32 by Espressif Systems**" după cum este ilustrat în Figura 8.

După finalizarea instalației, dați click pe selectorul de plăci din bara de sus, și alegeți "**select other board and port**".

În căsuța de dialog pentru selecția plăcilor, introduceți termenul de căutare "**doit**", apoi alegeți **DOIT ESP32 DEVKIT V1**.


File Edit Sketch Tools Help
Arduino Mega or Mega 2...
BOARDS MANAGER
2
ESP32
Type: All
1
Arduino ESP32 Boards by Arduino
Boards included in this package: Arduino Nano ESP32
More info
2.0.13
INSTALL
esp32 by Espressif Systems
Boards included in this package: ESP32 Dev Board, ESP32-S2 Dev Board, ESP32-S3 Dev Board, ESP32-C...
More info
2.0.11
INSTALL
3
sketch_nov
1
2
3
4
5
6
7
8
9
10

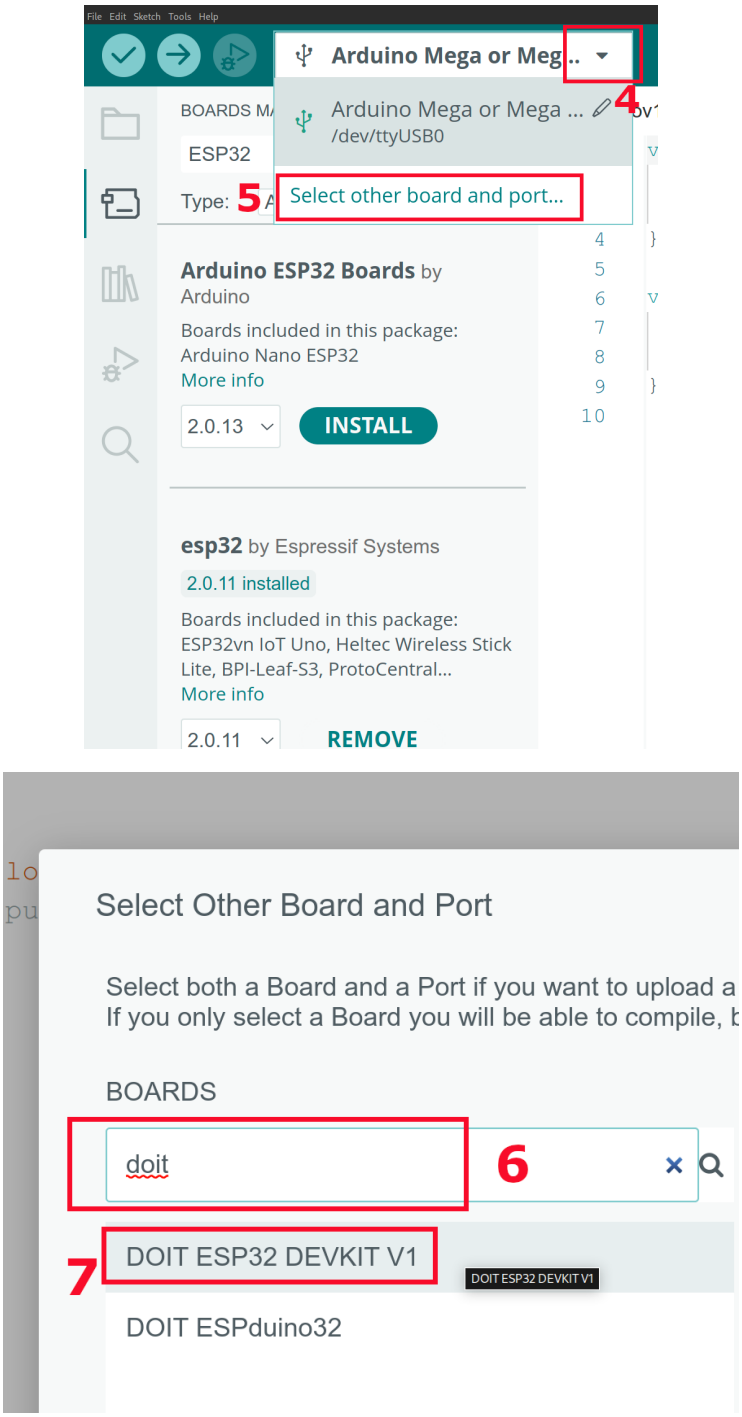


Figura 8. Configurarea mediului Arduino IDE v2.x. pentru plăcile ESP32.

4. Programarea plăcii ESP32

Conectați placa la PC folosind un **cablu Micro USB**. Selectați portul serial corect din meniul Tools al mediului Arduino IDE, și tipul plăcii (vezi capitolul anterior).

Tastați sau copiați programul dorit în mediul Arduino IDE, și folosiți aceeași comandă "Upload" pe care ați utilizat-o până acum pentru a compila și a scrie programul pe placă.

Pentru a testa funcționarea plăcii și a procesului de programare, puteți utiliza un program simplu Blink, care va aprinde și va stinge un LED montat pe placă, conectat la pinul digital 2.

```
void setup() {
  pinMode(2, OUTPUT);
}
void loop() {
  digitalWrite(2, 1);
  delay (500);
  digitalWrite(2,0);
  delay(500);
}
```

Procesul de programare a plăcii ESP32 durează mai mult decât procesul similar al plăcilor Arduino bazate pe AVR. Dacă apare eroarea **"Wrong boot mode detected (0x13)! The chip needs to be in download mode."** în timpul procesului de programare, trebuie să apăsați butonul BOOT de pe placa ESP32 după ce apare mesajul "Connecting…", și să îl țineți apăsat până ce apar mesajele "Writing…". După ce apar aceste mesaje, puteți elibera butonul BOOT, și să așteptați până ce procesul este complet.

Dacă procesul de programare este finalizat cu succes, consola ar trebui să arate ca în Figura 9.

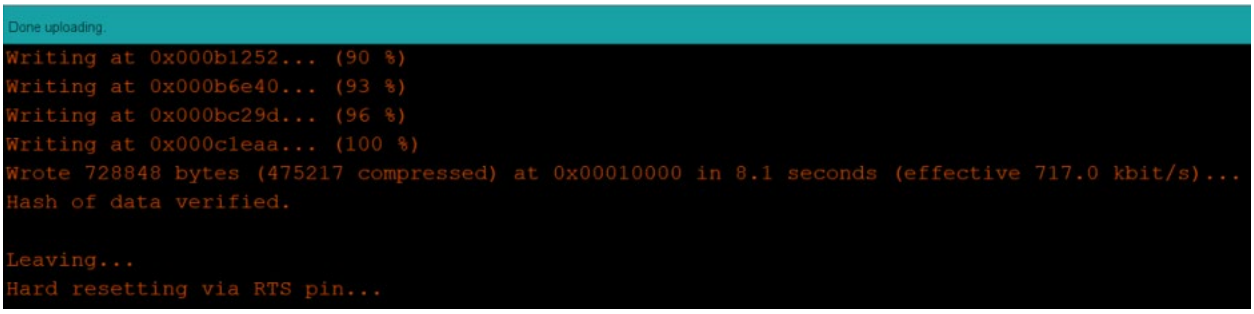


Figura 9. Mesajele din consolă după programarea corectă a ESP32.

Dacă programul dvs folosește interfața Serial pentru intrare/ieșire, dar după ce deschideți Serial Monitor nu apare nici un mesaj, este posibil să fie nevoie să apăsați butonul Reset.

5. Program demonstrativ pentru afișorul OLED

Pentru utilizarea afișorului OLED SSD1306, va trebui să instalați în prealabil o bibliotecă compatibilă cu acesta. Se recomandă utilizarea bibliotecii Adafruit SSD1306,

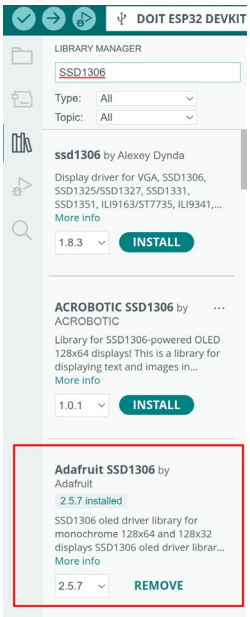


Figura 10. Instalarea bibliotecii “Adafruit SSD1306” din managerul de biblioteci al Arduino IDE.

Următorul cod demonstrativ poate fi descărcat de la adresa https://github.com/UTCN-AC-CS-DMP/Lab-7-ESP32-Part-1/blob/main/Lab_7_numai_OLED.ino

```
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define SCREEN_WIDTH 128  // Lățime afișor OLED, în pixeli
#define SCREEN_HEIGHT 32  // Înălțime afișor OLED, în pixeli

// Declararea afișorului SSD1306, conectat la pinii I2C (SDA, SCL)
// Pinii sunt definiți de biblioteca Wire.
// La Arduino UNO:      A4(SDA), A5(SCL)
// La Arduino MEGA:    20(SDA), 21(SCL)
// La ESP 32:      21 (SDA), 22 (SCL)
#define OLED_RESET -1
// Numărul pinului de reset (-1 dacă se folosește pinul reset al plăcii)
#define SCREEN_ADDRESS 0x3C
// Adresa I2C pentru afișorul OLED, 0x3D pentru 128x64, 0x3C pentru 128x32
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire,
                         OLED_RESET);


#define LOGO_WIDTH 128
#define LOGO_HEIGHT 32
// 'logo_utcn', 128x32px
static const unsigned char PROGMEM logo_bmp[] = {
 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xc0, 0x00, 0x3f,
 0x80, 0x00, 0x7f, 0x00, 0x00, 0xfe, 0x00, 0x01, 0xfc, 0x00, 0x03, 0xff,
 0xff, 0xfe, 0x07, 0xff, 0xfc, 0x0f, 0xff, 0xf8, 0x1f, 0xff, 0xf0, 0x3f,
 0xff, 0xe0, 0x7f, 0xff, 0xff, 0xc2, 0x04, 0x3f, 0x84, 0x08, 0x7f, 0x08,
 0x10, 0xfe, 0x10, 0x21, 0xfc, 0x20, 0x43, 0xff, 0xff, 0xc2, 0x04, 0x3f,
 0x84, 0x08, 0x7f, 0x08, 0x10, 0xfe, 0x10, 0x21, 0xfc, 0x20, 0x43, 0xff,
 0xff, 0xc2, 0x04, 0x3f, 0x84, 0x08, 0x7f, 0x08, 0x10, 0xfe, 0x10, 0x21,
 0xfc, 0x20, 0x43, 0xff, 0xff, 0xc2, 0x04, 0x3f, 0x84, 0x08, 0x7f, 0x08,
 0x10, 0xfe, 0x10, 0x21, 0xfc, 0x20, 0x43, 0xff, 0xff, 0xc2, 0x04, 0x3f,
 0x84, 0x08, 0x7f, 0x08, 0x10, 0xfe, 0x10, 0x21, 0xfc, 0x20, 0x43, 0xff,
 0xff, 0xe2, 0x04, 0x7f, 0xc4, 0x08, 0xff, 0x88, 0x11, 0xff, 0x10, 0x23,
 0xfe, 0x20, 0x47, 0xff, 0xff, 0xe2, 0x04, 0x7f, 0xc4, 0x08, 0xff, 0x88,
 0x11, 0xff, 0x10, 0x23, 0xfe, 0x20, 0x47, 0xff, 0xff, 0xf2, 0x04, 0xff,
 0xe4, 0x09, 0xff, 0xc8, 0x13, 0xff, 0x90, 0x27, 0xff, 0x20, 0x4f, 0xff,
 0xff, 0xfa, 0x05, 0xff, 0xf4, 0x0b, 0xff, 0xe8, 0x17, 0xff, 0xd0, 0x2f,
 0xff, 0xa0, 0x5f, 0xff, 0xff, 0xfe, 0x07, 0xff, 0xfc, 0x0f, 0xff, 0xf8,
 0x1f, 0xff, 0xf0, 0x3f, 0xff, 0xe0, 0x7f, 0xff, 0xff, 0xff, 0xff, 0xff,
 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xe1, 0xf1,
 0xff, 0xf8, 0x00, 0x7f, 0xff, 0xc0, 0xff, 0xff, 0x0f, 0x0f, 0xff, 0xff,
 0xff, 0xff, 0xe1, 0xf1, 0xff, 0xf8, 0x00, 0x7f, 0xff, 0x00, 0x7f, 0xff,
 0x07, 0x0f, 0xff, 0xff, 0xff, 0xff, 0xe1, 0xf1, 0xff, 0xff, 0x87, 0xff,
 0xfe, 0x1e, 0x7f, 0xff, 0x07, 0x0f, 0xff, 0xff, 0xff, 0xff, 0xe1, 0xf1,
 0xff, 0xff, 0x87, 0xff, 0xfe, 0x3f, 0x7f, 0xff, 0x03, 0x0f, 0xff, 0xff,
 0xff, 0xff, 0xe1, 0xf1, 0xff, 0xff, 0xff, 0x87, 0xff, 0xfc, 0x3f, 0xff, 0xff,
 0x01, 0x0f, 0xff, 0xff, 0xff, 0xff, 0xe1, 0xf1, 0xff, 0xff, 0x87, 0xff,
 0xfc, 0x7f, 0xff, 0xff, 0x11, 0x0f, 0xff, 0xff, 0xff, 0xff, 0xe1, 0xf1,
 0xff, 0xff, 0x87, 0xff, 0xfc, 0x7f, 0xff, 0xff, 0x10, 0x0f, 0xff, 0xff,
```

```
 0xff, 0xff, 0xe1, 0xf1, 0xff, 0xff, 0x87, 0xff, 0xfc, 0x7f, 0xff, 0xff,
 0x18, 0x0f, 0xff, 0xff, 0xff, 0xff, 0xf1, 0xf1, 0xff, 0xff, 0x87, 0xff,
 0xfc, 0x3f, 0xff, 0xff, 0x1c, 0x0f, 0xff, 0xff, 0xff, 0xff, 0xf1, 0xf1,
 0xff, 0xff, 0x87, 0xff, 0xfc, 0x3f, 0x7f, 0xff, 0x1c, 0x0f, 0xff, 0xff,
 0xff, 0xff, 0xf0, 0xe1, 0xff, 0xff, 0x87, 0xff, 0xfe, 0x1e, 0x7f, 0xff,
 0x1e, 0x0f, 0xff, 0xff, 0xff, 0xff, 0xf8, 0x03, 0xff, 0xff, 0x87, 0xff,
 0xff, 0x00, 0x7f, 0xff, 0x1e, 0x0f, 0xff, 0xff, 0xff, 0xff, 0xfc, 0x0f,
 0xff, 0xff, 0x87, 0xff, 0xff, 0xc0, 0xff, 0xff, 0x1f, 0x0f, 0xff, 0xff,
 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff
};


void setup() {
 Serial.begin(9600);
 // SSD1306_SWITCHCAPVCC = generare tensiune pentru afișaj din tensiunea internă de 3.3V
 if (!display.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS)) {
   Serial.println(F("SSD1306 allocation failed"));
   for (;;)
     ;  // Nu continua, blochează programul aici
 }


 // Șterge afișaj
 display.clearDisplay();


 // Desenează logo demo
 testDrawUTCNLogo();


 // Scrie șiruri de caractere pe rânduri diferite
 writeTextSSD1306("First line", 0, 1, true);
 writeTextSSD1306("Another string below", 8, 1, false);
 // Întârziere 3 secunde
 delay(3000);
 // Text cu dimensiune mai mare, se trece automat la linia următoare
 writeTextSSD1306("Auto line break", 0, 2, true);

 delay(3000);

 display.setCursor(0, 0);
 display.clearDisplay();

 display.write("Waiting");
}

int c = 0;

void loop() {

 display.write(".");
 display.display();

 delay(1000);

 c++;

 if (c > 13) {
   display.clearDisplay();
   display.setCursor(0, 0);
```

```
    c = 0;
  }
}


void writeTextSSD1306(const String &message, const uint8_t rowNum, const uint8_t textSize,
const bool clearDisp)
{
 if (clearDisp) {
   display.clearDisplay();
 }

 display.setTextSize(textSize);
 display.setTextColor(SSD1306_WHITE);  // Desenează text alb
 // Specifică poziția mesajului
 display.setCursor(0, rowNum);
 // Folosește fontul complet pentru 256 caractere 'Code Page 437'
 display.cp437(true);
 display.write(message.c_str());
 display.display();
}

void testDrawUTCNLogo(void) {
 display.clearDisplay();


 display.drawBitmap(0, 0, logo_bmp, LOGO_WIDTH, LOGO_HEIGHT, WHITE);
 display.display();
 delay(3000);
}
```

Elemente importante: programul este scris pe baza obiectului **display**, din clasa **Adafruit_SSD1306**. Constructorul clasei primește ca parametri lățimea și înălțimea afișorului, și un pointer către obiectul Wire folosit pentru comunicarea I2C. Inițializarea obiectului display primește ca parametru adresa I2C a afișorului, care este 0x3C.

Obiectul display permite ștergerea folosind metoda **clearDisplay**, poate poziționa cursorul pentru scrierea de text folosind **setCursor**, poate configura dimensiunea textului folosind **setTextSize**, și poate scrie un text folosind **write**. O imagine grafică poate fi afișată folosind metoda **drawBitmap**.

6. Program demonstrativ pentru WiFi (ESP32 ca punct de acces)

Următorul cod demonstrativ poate fi descărcat de la adresa https://github.com/UTCN-AC-CS-DMP/Lab-7-ESP32-Part-1/blob/main/Lab_7_numai_WiFi.ino

```
#include <WiFi.h>
#include <WiFiAP.h>
#include <WiFiClient.h>

// Șiruri de caractere cu mesaje predefinite
```

```
const String SETUP_INIT = "SETUP: Initializing ESP32 dev board";
const String SETUP_ERROR = "!!ERROR!! SETUP: Unable to start SoftAP mode";
const String SETUP_SERVER_START = "SETUP: HTTP server started --> IP addr: ";
const String SETUP_SERVER_PORT = " on port: ";
const String INFO_NEW_CLIENT = "New client connected";
const String INFO_DISCONNECT_CLIENT = "Client disconnected";

// Un header HTTP începe întotdeauna cu un cod de răspuns (e.g. HTTP/1.1 200 OK)
// și tipul conținutului, pentru a informa clientul, urmat de o linie goală:
const String HTTP_HEADER = "HTTP/1.1 200 OK\r\nContent-type:text/html\r\n\r\n";
const String HTML_WELCOME = "<h1>Welcome to your ESP32 Web Server!</h1>";

// Constante pentru configurarea de bază a WIFI
// SSID (Service Set IDentifier), numele rețelei
const char *SSID = "<your_unique_SSID>";
// Parola pentru rețea
// Implicit, ESP32 foloseste modul WPA/WPA2
// astfel că parola trebuie să aibă între 8 și 63 caractere ASCII
const char *PASS = "<your_password_here>";
// Portul implicit pentru un server HTTP este 80, conform RFC1340
const int HTTP_PORT_NO = 80;

// Initializare server HTTP pe portul 80
WiFiServer HttpServer(HTTP_PORT_NO);

void setup() {
 Serial.begin(9600);

 if (!WiFi.softAP(SSID)) {
       // înlocuiți cu if (!WiFi.softAP(SSID, PASS)) pentru a utiliza parola
   Serial.println(SETUP_ERROR);
   // Dacă nu se poate activa punctul de acces, blochează programul aici
   while (1)
     ;
 }

 // Citire adresă IP a AP-ului pentru mesaj de informare
 const IPAddress accessPointIP = WiFi.softAPIP();
 const String webServerInfoMessage = SETUP_SERVER_START + accessPointIP.toString()
                                     + SETUP_SERVER_PORT + HTTP_PORT_NO;

 // Pornire server HTTP
 HttpServer.begin();
 Serial.println(webServerInfoMessage);
}

void loop() {
 WiFiClient client = HttpServer.available();  // Ascultă pentru clienți noi

 if (client) {                          // dacă avem un client conectat,
   Serial.println(INFO_NEW_CLIENT);     // trimite un mesaj pe portul serial
   String currentLine = "";            // Șir pentru a citi datele de la client
   while (client.connected()) {         // cât timp clientul este conectat
     if (client.available()) {          // dacă avem caractere de citit de la client,
       const char c = client.read();    // citește un caracter, apoi
       Serial.write(c);                 // tipărește la serial monitor

       if (c == '\n') {  // dacă caracterul este new line
         // dacă linia este goală, avem două caractere newline consecutive
         // asta înseamnă finalul cererii HTTP de la client, deci trimitem răspuns:
         if (currentLine.length() == 0) {
           // Trimite mesaj de bun venit
```

```
            printWelcomePage(client);
            break;
          } else currentLine = "";
        } else if (c != '\r') {  // dacă există alte caractere în afară de carriage return
          currentLine += c;      // se adaugă la linia curentă
        }
      }
    }
    // se închide conexiunea:
    client.stop();
    Serial.println(INFO_DISCONNECT_CLIENT);
    Serial.println();
  }
}

void printWelcomePage(WiFiClient client) {
  // Răspunsul către client trebuie să conțină headerele corecte
  client.println(HTTP_HEADER);

  // Trimitem mesajul HTML
  client.print(HTML_WELCOME);

  // Răspunsul HTTP se termină cu o linie goală
  client.println();
}
```

Elemente importante și avertizări: orice access point WiFi are un identificator unic, **SSID**-ul, și poate fi securizat cu o parolă. Vă rog să vă asigurați că veți configura un SSID care este unic pentru voi, și poate fi ușor identificat, deoarece două SSID-uri identice pot cauza probleme în rețea. Vă rog să modificați linia următoare pentru a vă asigura că SSID-ul vostru este unic:

```
const char *SSID = "<your_unique_SSID>";
```

Conexiunile sunt tratate de către obiectul **HttpServer**, din clasa **WiFiServer**. Serverul ascultă pe portul specificat (portul 80 pentru protocolul HTTP) și va returna un obiect **client**, din clasa **WiFiClient**, atunci când un dispozitiv precum un laptop sau un telefon mobil se conectează (prin deschiderea unui browser și tastarea adresei serverului). Obiectul **client** va citi datele de la dispozitivul conectat folosind metode similare cu cele ale interfeței Serial (**available**, **read**), și poate de asemenea să transmită date folosind metodele similare (**write**, **print**, **println**). Comunicarea dintre ESP32 și dispozitivul client este deci o comunicare de tip text simplu. Programatorul este responsabil pentru formatarea textului de ieșire pentru a forma o pagină html validă, și de interpretarea textului de intrare pentru a identifica cererile de la dispozitivul client.

Pentru a testa programul trebuie să în încărcați pe placa ESP32, și apoi să configurați dispozitivul client (laptop, telefon) să se conecteze la access point-ul WiFi al plăcii ESP32. Acest punct de acces va fi identificat prin SSID-ul unic. După ce ați configurat conexiunea, veți deschide browserul și veți tasta adresa de IP a plăcii, care va fi 192.168.4.1. Serverul ar trebui să răspundă cu un mesaj de întâmpinare, ca în Figura 11.

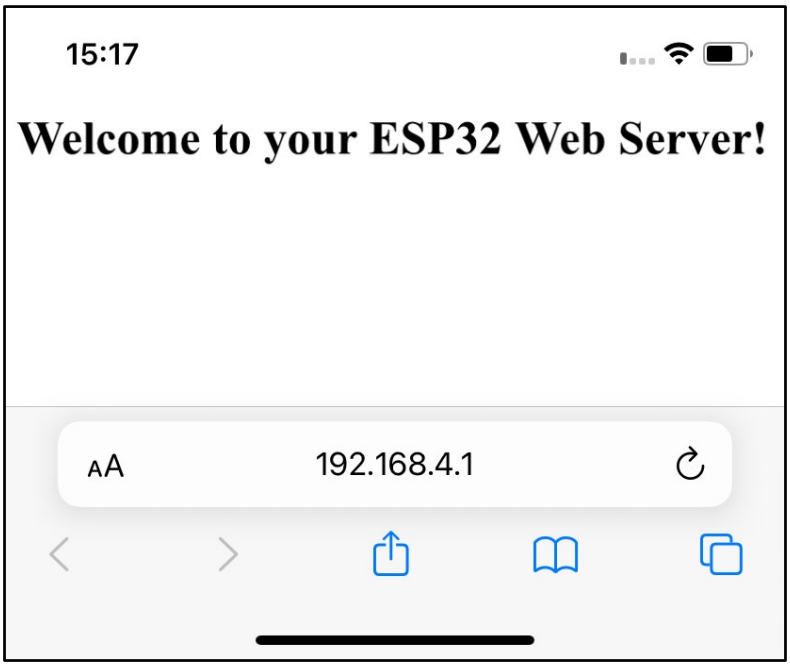


Figura 11. Pagina web transmisă de serverul de web al ESP32.

NOTĂ: Dacă dețineți Apple iPhone și utilizați browserul Safari, trebuie să specificați și protocolul folosit, nu doar adresa IP, pentru că Safari va încerca să folosească implicit HTTPS. Astfel, veți tasta http://192.168.4.1 în bara de adresă, nu doar 192.168.4.1.

7. Lucru individual

1. Rulați exemplele, analizați codul și explicațiile.
2. Afișați starea conexiunii și șirurile de tip cerere GET primite de serverul web WiFi pe afișorul OLED.
3. Pe baza cererilor GET, aprindeți sau stingeți LED-ul plăcii (*indiciu: puteți adăuga text după adresa IP, de exemplu 192.168.4.1/LedOn*)
4. Modificați serverul web pentru ca pagina html transmisă să conțină linkuri pentru aprinderea sau stingerea LED-ului.

Bibliografie

1. Descrierea microcontrollerului ESP32 https://www.espressif.com/en/products/socs/esp32

2. API-ul ESP32 WiFi https://docs.espressif.com/projects/arduino-esp32/en/latest/api/wifi.html

3. Datasheet-ul afișorului SSD1306 https://cdn-shop.adafruit.com/datasheets/SSD1306.pdf

4. Documentația YwRobot https://static.rapidonline.com/pdf/73-4538_v1.pdf

5. Descriere breadboard https://www.adafruit.com/product/64