ESP32 WiFi Provisioning

Maj Rakovec

Ljubljana, Slovenia
maj.rakovec@icloud.com

***Abstract*—ESP32 Access Point, accepting credentials of existing WiFi networks as an input, processing the input and connecting to the specified network.**

Keywords—ESP32, Access Point, WiFi Provisioning, Arduino IDE, server, EEPROM, IP address

Requirements—ESP32 development board, Arduino IDE or PlatformIO with ESP32 support, WiFi.h, EEPROM.h and ESPAsyncWebServer.h library

I. Introduction

The ESP32 WiFi Provisioning web project, referred throughout this document as the ESP32 project, aims to configure and connect a device, such as an IoT device or a microcontroller like ESP32, to a WiFi network.

When started/restarted, the ESP32 first acts as an access point or AP, enabling other devices to connect to its own network. It hosts a local webpage on a specified IP address, where a user can input the credentials of a desired WiFi network. The ESP device then proceeds with a self-restart and lastly it connects to the network specified by the user.

The following sections of this proposal describe the development process and functionalities of each step.

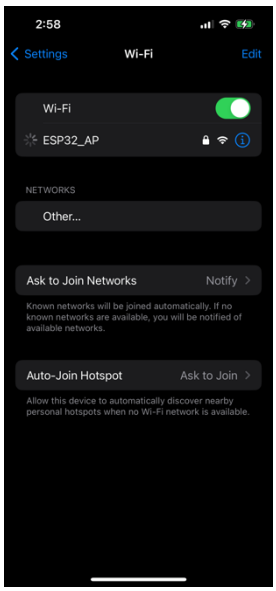
II. Usage

Connecting an IoT device or a microcontroller (ESP32) to a WiFi network through the device's AP mode.

III. Functionalities

A. WiFi Provisioning

When the ESP32 starts up, it sets itself in AP mode and creates a WiFi network with the SSID "ESP32-Access-Point" and a default password "123456789." Users can connect their devices (e.g., computer or smartphone) to this WiFi network.



B. Web Server:

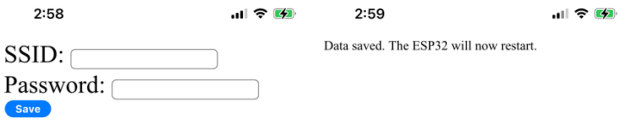
Once connected to the ESP32's WiFi network, users can access a web page hosted by the ESP32. This web page is loaded in the user's web browser and serves as the interface for the WiFi provisioning process. The webpage's name should be the same as the ESP's local IP address (e.g. http://192.168.4.1/).

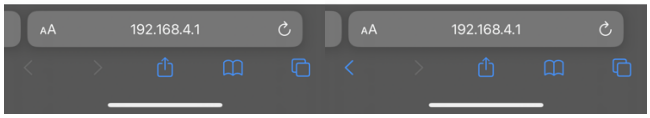
C. WiFi Network Credentials Input

The web page presents two input boxes where users can input their WiFi network credentials - SSID (network name) and password. The input boxes are followed by a "Save" button which, when clicked, saves the credentials on the ESP32 board.

D. Saving WiFi Credentials

When users input their WiFi network credentials and click the "Save" button, the ESP32 captures the data and saves it in non-volatile memory, thanks to the EEPROM library. The credentials will be used for connecting to the WiFi network on subsequent restarts.

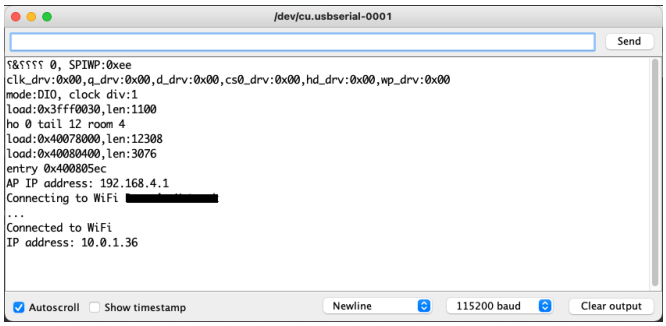




E. Restart and Connect

After successfully receiving the WiFi network credentials, the ESP32 restarts, exits AP mode. It then connects to the WiFi network using the previously provided credentials.

In case the device wasn't able to connect to the specified network, it will set the device back to the AP mode and wait for another user input.



IV. Code Breakdown

A. *Library Inclusions*

The libraries necessary for the code to function properly are already included in the code. Those are <WiFi.h>, <EEPROM.h> and < ESPAsyncWebServer.h>.

B. *Configuration*

In the first lines of code, SSID and password of ESP32 AP mode are already specified. If desired, the user can change this section, as it has no further effect on the code. These values are the login credentials of the ESP32 device and will be used in both AP mode and WiFi client mode.

C. *EEPROM Address Definitions*

The last lines before the setup function define the addresses where the SSID and password will be stored in the ESP's EEPROM.

D. *Setup Function*

- Standard configuration of serial port for communication and debugging.
- Set the ESP in AP mode and display the local IP address in serial port
- Read stored SSID and password from EEPROM (if there are any). If any valid credentials are found, the ESP32 tries connecting to the WiFi network.

E. *Setup Function*

In this project, the loop function can be left empty. It still needs to exist in the code, as it is a requirement for Arduino/ESP code to function.

F. *Web Server Setup*

In this part, the root path ("/") request handlers are defined to store or save it ("/save"). On the root path, a web form is served to allow users to input their WiFi credentials and on the save path, the form data is processed, and the credentials are stored in the ESP's EEPROM.

G. *Additional Functions*

readFromEEPROM and writeToEEPROM functions have fairly self-explanatory names. The first one reads data from the ESP's EEPROM (it reads from the previously specified address) and the latter writes new data on it.

The last function, connectToWiFi, serves the code to connect the device to the WiFi network using the stored credentials.

V. Notes

The ESP32 will automatically connect to the WiFi network (if found) on each restart, using the last stored credentials. To change the WiFi network, repeat the provisioning process.

The EEPROM also has a limited number of write cycles, so it's smart to try to avoid frequently changing the WiFi credentials.

VI. Troubleshooting

If you encounter problems uploading the code to the ESP32 device, make sure that the board is properly connected to your computer (check for faulty cables or connectors).

Verify that the Arduino IDE has the correct board package installed for the ESP32 and that you have all the necessary libraries installed.

If the ESP32 fails to connect to your WiFi network, consider checking the network credentials again.

VII. Summary

The project offers significant benefits, especially for Internet of Things (IoT) applications, where deploying and configuring multiple devices efficiently is crucial. WiFi provisioning allows users to set up their ESP32 devices with ease, without the need for direct physical access to each device or the hassle of manually hardcoding WiFi credentials into the code.

The project's flexibility, convenience, and simplicity make it a valuable tool for developers and makers looking to create IoT projects using ESP32 microcontrollers. Additionally, the web-based provisioning approach ensures that the ESP32 remains versatile and can be adapted to various network environments and user preferences.

VIII. Conclusions

The "ESP32 WiFi Provisioning Web Server" project simplifies the process of configuring and connecting an ESP32 to a WiFi network. Through the web-based interface, users can easily input their WiFi network credentials, enabling the ESP32 to connect to the desired network without the need for hardcoding credentials in the code. This project is particularly useful for IoT applications where easy deployment and configuration are essential.

IX. Alternatives

Another way of getting an ESP32 to go through the AP mode to connect to a desired WiFi network would be with a library called WiFiManager. It offers similar functionalities as the code provided in this project. The creators of the library also offer user support and code examples on their GitHub page.

References And Resources

[1] ESPAsyncWebServer library – GitHub: https://github.com/me-no-dev/ESPAsyncWebServer

[2] EEPROM library – GitHub: https://github.com/PaulStoffregen/EEPROM

[3] WiFiManager library – GitHub: https://github.com/tzapu/WiFiManager

[4] uPesy: How to connect to a WiFi network with the ESP32: https://www.upesy.com/blogs/tutorials/how-to-connect-wifi-acces-point-with-esp32#

[5] Random Nerd Tutorials: Input Data on HTML Form ESP32/ESP8266 Web Server using Arduino IDE: https://randomnerdtutorials.com/esp32-esp8266-input-data-html-form/

[6] Random Nerd Tutorials: How to Set an ESP32 Access Point (AP) for Web Server: https://randomnerdtutorials.com/esp32-access-point-ap-web-server/#:~:text=Connecting%20to%20the%20ESP32%20Access%20Point&text=Open%20your%20web%20browser%20and,ESP32%2DAccess%2DPoint%E2%80%9C

[7] ESP8266 Community Forum: https://www.esp8266.com/