Web Configurator for Technical Devices using ESP32 and Captive Portal with Persistent Storage

Simón Aulet

Problem: Traditional Configuration Interfaces

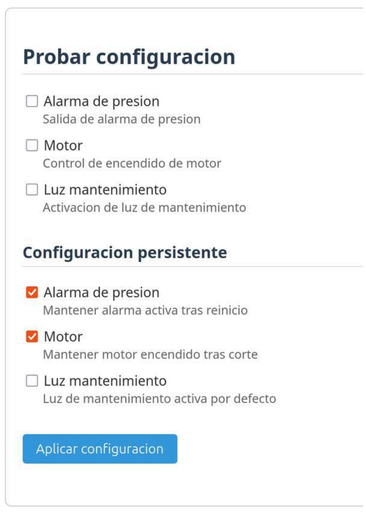
Most technical or industrial devices include small displays and buttons to allow parameter configuration. This approach presents several disadvantages:

- The interface is limited in space and visibility.
- Entering data or navigating through multiple options is often slow and prone to errors.
- In environments with low lighting, limited physical access, or the need for frequent changes, this method becomes especially inconvenient.

The image shows real examples of display-based configurators, where navigation typically relies on sequential menus with physical buttons and without clear feedback.

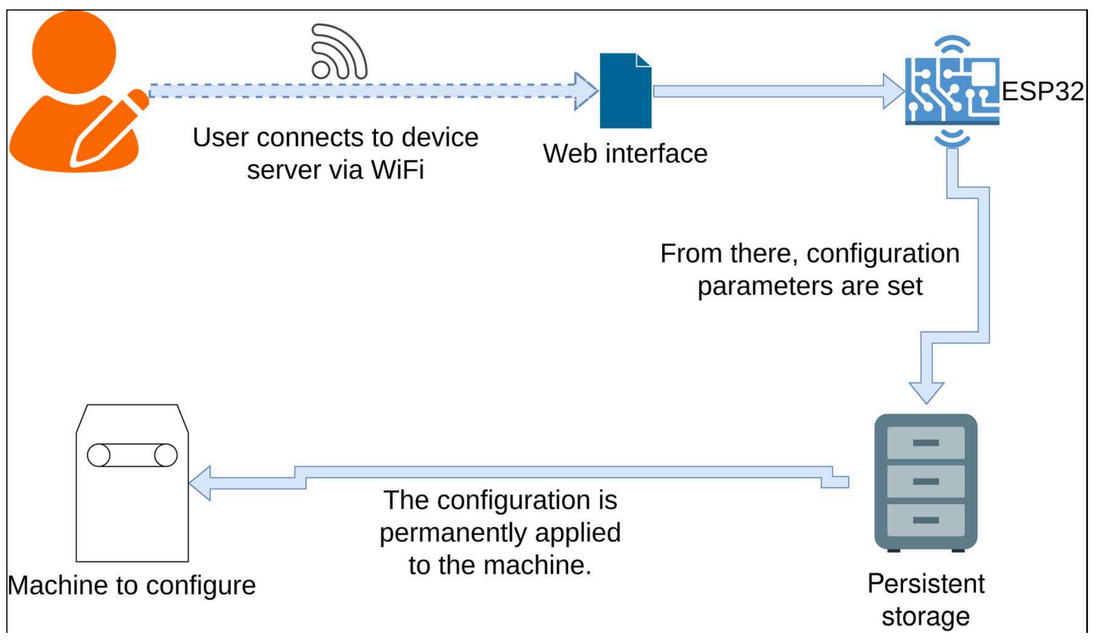
Solution: Web WiFi interface using ESP32

As a modern and more efficient alternative, the proposal is to use an ESP32 acting as a Wi-Fi access point. Upon connecting, the user is automatically redirected to a web interface hosted directly on the device. From there, they can view and modify the configuration in a clear and intuitive way, without the need for physical displays or buttons.
This approach significantly improves the user experience, simplifies technical maintenance, and allows the system to be easily extended to different types of equipment.
Additionally, the configurations are stored in persistent memory (NVS), ensuring they remain intact even after resets or power outages.



Working principle

The user accesses the system by connecting via Wi-Fi to a network created by the ESP32 microcontroller. Upon doing so, they are automatically redirected to a web page stored locally on the device itself. This web interface allows users to view the current status of the equipment parameters and modify its configuration in an intuitive and accessible way from any browser. Once the desired adjustments are made, the values are stored in the microcontroller's non-volatile memory, ensuring that the configuration is preserved even after resets or power disconnections. This configuration is then directly applied to the machine or system being controlled through the ESP32's GPIO pins, which enable or disable specific functions according to the user's settings.



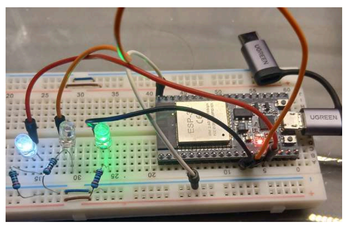
Tests

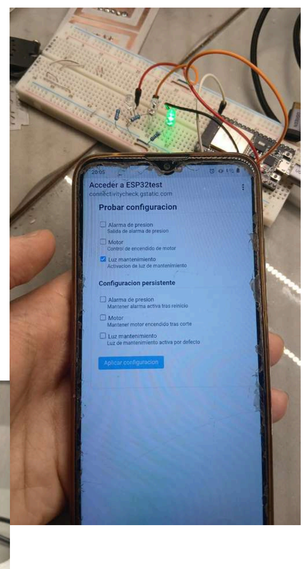
Hardware verification:

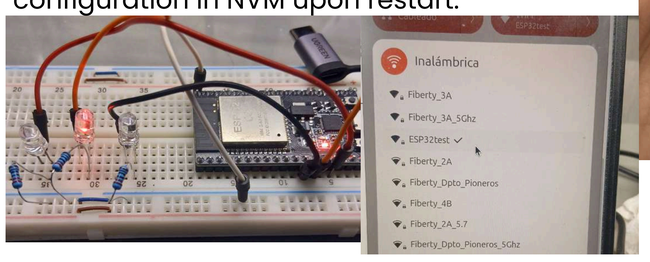
- LEDs connected to GPIO pins (12, 13, 14).
- Control of outputs from the web interface via serial connection.
- Verification of real states on the prototype when modifying the web interface.

System validation:

- Successful Wi-Fi connection from different devices.
- Automatic redirection to the web interface (captive portal).
- Correct R/W operations in NVM memory.
- Automatic application of previously stored configuration in NVM upon restart.







Conclusiónes y mejoras futuras

The developed system effectively solved the problem of technical configuration in embedded devices, eliminating the need for complex physical interfaces.
The solution is portable, cost-effective, and adaptable to multiple applications.

Future improvements:

- Incorporate different types of possible configurations (I2C communication, ADC for reading additional sensors, etc.).
- External network connectivity (Wi-Fi client mode) for automated reporting.
- Security through authentication in the portal.
- Communication with other microcontrollers for more complex machines that require more than a single ESP32.

Resources

Project repository:
https://github.com/SimonAulet/Configurador-ACySE

Software used for this project

- ESP-IDF
- Sublime Text + Terminal
- Sublime merge
- Draw.io

Hardware used for this project

- ESP32 WROOM
- Protoboard
- 3 LED diodes
- 3 220 ohm resistors
- Wires for connections

Schematic