



NVS Implementation

Saving Data Using the ESP-IDF NVS Library

The background features a complex network of thin, light-colored lines and small circles, forming a web-like structure. Overlaid on this are several translucent triangles of various sizes and orientations. The overall aesthetic is technical and modern, with a color palette dominated by light greys, blues, and greens.

Our Implementation

Saving, Loading, Clearing WiFi Credentials

- About the Implementation

- After successfully connecting the ESP32 to an access point via the web page, the SSID and Password used to connect will be saved to flash.
- Upon startup, the ESP32 will check the flash for any saved credentials, if there are any, they will be used to attempt a connection immediately.
- Upon startup, if after the MAX_CONNECTION_RETRIES is reached and a connection cannot be established, we will clear the flash.
- If the “Disconnect Button” on the web page is pressed, the credentials will be cleared from the flash.

The background features a complex network of thin, light-colored lines and small circles, forming a web-like structure. Overlaid on this are several translucent triangles of various sizes and orientations, some of which are filled with a light yellow or orange hue. The overall aesthetic is technical and modern.

ESP-IDF NVS Library APIs Used, in Brief

Utilizing ESP-IDF for NVS

- Suggested Reading & About

- About the Non-volatile Storage Library and API Reference → https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/storage/nvs_flash.html
 - NVS components save and load data from storage that is preserved between boots.
 - NVS works best for storing many small values e.g., WiFi credentials in our case.
 - Keys and Values → Keys are ASCII strings and values are the types (variable length binary data (blob) in our case)
 - Namespaces → We assign each key-value pair to a namespace, and we specify this namespace when using certain NVS APIs.
-

NVS for WiFi Credentials Using the ESP-IDF

- ESP IDF APIs Used
 - Saving the credentials
 - Open storage area with a given namespace → Call [nvs_open](#) and specify the namespace name, open in Read/Write (in our case) and we need to pass a handle as well which will be used for subsequent calls to [nvs_set_*](#), and [nvs_commit](#) functions.
 - Set variable length binary value (SSID & Password) → Using [nvs_set_blob](#).
 - After setting the values, we need to write the changes to NVS → Using [nvs_commit](#).
 - Getting the credentials
 - Similarly, making changes, when retrieving information from the flash, we must open using [nvs_open](#), then, in our case → Call [nvs_get_blob](#).
-

NVS for WiFi Credentials Using the ESP-IDF...

- ESP IDF APIs Used
 - Clearing the credentials
 - Similarly, we open storage area with a given namespace → Call [nvs_open](#) and specify the namespace name, open in Read/Write and we need to pass the handle.
 - Now, we erase the key-value pairs in the namespace → Using [nvs_erase_all](#).
 - After erasing, we need to commit the changes to NVS → Using [nvs_commit](#).
-

The background features a complex network of thin, light-colored lines and small circles, creating a web-like or molecular structure. Scattered throughout are various triangles of different sizes and orientations, some of which are filled with a light blue or yellow color. The overall aesthetic is clean, modern, and technical.

Let's get started!