

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

WiFi Connect Implementation

Connecting the ESP32 to an Access Point

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 clean, modern, and technical.

Our Implementation

WiFi Connection

- **About the Implementation**

- Credentials should be entered into text fields of the web page, where a “connect” button is available to kick off the connection process.
- The web server will handle receiving the credentials and the application will update the WiFi configuration and then attempt the connection.
- The application will let the user know by displaying the connection result on the web page; success for failure.
- Connection details (AP name and IP information) will also be displayed on the web page (when there is a connection).
- A “disconnect” button will also be available to the user for disconnecting from an active connection.
- The RGB LED should indicate a connected status.

The background features a complex network of thin, light-colored lines connecting small circular nodes, creating a web-like or molecular structure. Scattered throughout are various geometric shapes, including triangles and polygons, some of which are filled with a light yellow or orange color. The overall aesthetic is technical and modern.

ESP-IDF WiFi Driver APIs Used for WiFi Connection, in Brief

Making a WiFi Connection Using the ESP-IDF

- ESP IDF APIs Used

- Within the URI Handler for receiving credentials → Get header value length, using [http_req_get_hdr_value_len](#).
 - Next, get the value string of a field from the request headers → [httpd_req_get_hdr_value_str](#).
 - Next, we need to update the WiFi configuration → Update the **station** configuration data in the union [wifi_config_t](#).
 - Set the configuration → For station mode [esp_wifi_set_config](#) the interface should be specified as [ESP_IF_WIFI_STA](#).
 - Attempt the connection → Using [esp_wifi_connect](#).
 - Disconnect from an active connection → Using [esp_wifi_disconnect](#).
-

Making a WiFi Connection (ESP-IDF), Continued...

- ESP IDF APIs Used (Displaying Connection Information to the Web Page...)
 - Get the access point information (SSID) for display on the page → Using wifi_ap_record_t and pass it to esp_wifi_sta_get_ap_info.
 - Get the IP connection information (assigned to the ESP32) for display on the page → Using esp_netif_ip_info_t and pass it to esp_netif_get_ip_info.
 - Convert the numeric IP address into dotted decimal ASCII → Using esp_ip4addr_ntoa.
-

WiFi Application

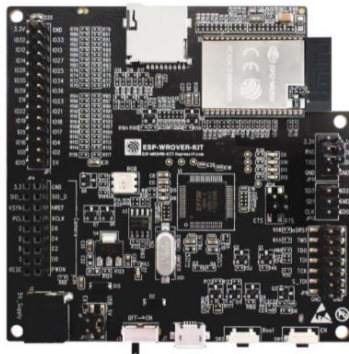
ESP32 SoftAP

- AP/STA combination mode.
- We assign an IP to the SoftAP; the interface of the ESP32 is statically configured.
- DHCP server dynamically assigns an IP for connecting stations.
- We set a maximum number of stations allowed to connect.



Connecting Device

- When the ESP32 connects to an AP, local time can be obtained utilizing SNTP (if the AP is connected to the internet).
- DHCP service from the AP, will dynamically assign an IP to the ESP32 in our application.



Connecting Device

- Becomes "station" of ESP32's SoftAP when connected.
- DHCP service from the ESP32's SoftAP will dynamically assign an IP to your device.
- Interact with the ESP32 via the web page.



The background features a complex network of thin, light-colored lines and small circles, creating a web-like structure. Overlaid on this are several triangles of various sizes and orientations, some of which are filled with a light blue or yellow color. The overall aesthetic is clean and modern, with a focus on geometric shapes.

Let's get started!