WiFi Implementation

WLAN (Wireless Local Area Network Application)

Our Implementation

WiFi Application (High-Level Perspective)

- About the Implementation
  - The ESP32 should start its Access Point so that other devices can connect to it.
    - This enables users to access information e.g., sensor data, device info., connection status/information, user option to connect to and disconnect from an AP, display local time, etc..
  - The WiFi application will start an HTTP Server (created in the next section), which will support a web page (to do all the stuff mentioned above).
  - The application will contain a FreeRTOS task that accepts FreeRTOS Queue messages (xQueueCreate, xQueueSend and xQueueReceive) for event coordination.
  - To be Implemented Later in the Course → connect the ESP32 using previously used (saved) credentials.

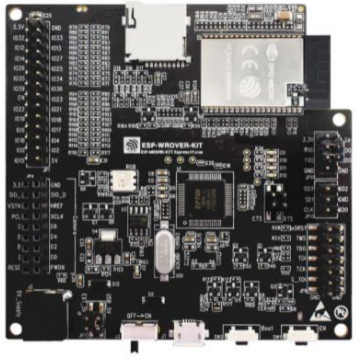
Simplified Overview of WiFi Application

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.

ESP32 SoftAP/Station

- 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.

ESP-IDF WiFi Driver, in Brief

Utilizing WiFi Driver From ESP-IDF

- Suggested Reading
  - WiFi Driver → https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-guides/wifi.html
  - API Reference → https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/network/esp_wifi.html
  - ESP-NETIF → https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/network/esp_netif.html

Notable ESP-IDF Functions Used in This Section

- Configuration Steps and ESP-IDF Functions Used
  - Define WiFi settings → Header file with SSID, Password, IP, Gateway, Netmask. etc.
  - Define WiFi FreeRTOS task → Use xTaskCreatePinnedToCore or xTaskCreate.
  - Create an event handler → Call esp_event_handler_instance_register.
  - Implement default configuration → Initialize TCP/IP stack using esp_netif_init and WiFi configuration by calling esp_wifi_init, esp_wifi_set_storage, and default configurations esp_netif_create_default_wifi_ap , esp_netif_create_default_wifi_sta.
  - Define ESP32 SoftAP configuration → Define AP Settings wifi_config_t struct and static IP (in our case). Functions Used esp_netif_set_ip_info, esp_netif_dhcps_start, esp_wifi_set_mode, esp_wifi_set_config, esp_wifi_set_bandwidth, esp_wifi_set_ps.
  - Start WiFi (esp_wifi_start).

Let’s get started!