HTTP Server Implementation

Key Component of the WLAN Application, Supporting the Web Page Functionality

Our Implementation

HTTP Server Requirements

- About the Implementation
  - The HTTP Server will support the web page files (.html, .css and .js).
  - It will also support OTA (Over the Air) Firmware Updates.
  - Additionally, support for DHT22 Temperature and Humidity Sensor readings for display on the web page will be added.
  - The HTTP server will be able to respond to Connection and Disconnection buttons on the web page e.g., by entering SSID & Password into text fields and clicking connect and disconnect for removing a connection.
  - The web server will also handle sending connection information (SSID and IP, Gateway, Netmask) about the active connection to the web page.
  - We will also send the ESP32's assigned SSID to the web page.

Web Page Files

- About the Files
  - index.html → HTML markup that we will expand upon (it already includes OTA Update markup).
  - app.css → Used to style for the index.html document that we will expand (already includes OTA Update styling).
  - app.js → Javascript file that we will spend time expanding throughout the course. The template provided already includes the OTA Update functions.
  - favicon.ico → Icon that gets displayed in the address bar of the browser.
  - jquery-3.3.1.min.js → JavaScript library.

HTTP Server Component from ESP-IDF, in Brief

Utilizing HTTP Server Component

- Suggested Reading
  - About HTTP Server and API Reference → https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/protocols/esp_http_server.html
  - Browse this material, as this example directly relates to our implementation.

Creating an HTTP Server Using the ESP-IDF

- Configuration Steps and Notable ESP-IDF Functions Used
  - Embed Binary Data (index.html, app.css and code.js) → Embedding Binary Data.
  - Create HTTP Server start and stop functions → create a wrappers around these.
  - Create the default HTTP server configuration and adjust to our needs → Create the struct httpd_config_t and call httpd_start.
  - Register the URI handlers → The first lesson, this will include the .html, .css and .js files) and call httpd_register_uri_handler for each.

Note: We will continue to expand the web server and URI handlers based on what we need to accomplish.

- *Also, we will have a "monitor" task which can receive queue messages to respond to certain events (similar to the WiFi application).*

Let’s get started!