OTA Firmware Update Implementation

Firmware Update Over WiFi, Update Using the Web Page

About OTA Updates

- Overview
  - Allows a device to update itself based on data received (for example, over WiFi) while the normal firmware is running.
  - Requires configurating a Partition Table of the device with at least two “OTA app slots” (i.e., ota_0 and ota_1) and an “OTA Data Partition”.
  - The OTA operation functions write a new app firmware image to whichever OTA app slot that is currently not selected for booting…
    - Once the image is verified, the OTA Data partition is updated to specify that this image should be used for the next boot.

About Partition Tables

- Overview
  - A single ESP32's flash can contain multiple apps, as well as many kinds of data (calibration data, filesystems, parameter storage, etc.), for this reason a partition table is flashed to 0x8000 (default offset) in the flash.
  - Each entry in the partition table has a name (label), type (app, data, etc.), subtype and the offset in flash where the partition is loaded.
  - The simplest way to use the partition table is to open the project configuration (sdkconfig in Eclipse) and choose "Factory app, two OTA definitions" (as we have already done in the project configuration.
  - The table is located: esp-idf\components\partition_table

More About Partition Tables

- Description of “Factory app, two OTA definitions” configuration:

```
# ESP-IDF Partition Table
# Name,   Type, SubType, Offset,  Size, Flags
nvs,      data, nvs,     0x9000,  0x4000,
otadata,  data, ota,     0xd000,  0x2000,
phy_init, data, phy,     0xf000,  0x1000,
factory,  app,  factory, 0x10000,  1M,
ota_0,    app,  ota_0,   0x110000, 1M,
ota_1,    app,  ota_1,   0x210000, 1M,
```

  - There are now three app partition definitions. The type of the factory app (at 0x10000) and the next two “OTA” apps are all set to “app”, but their subtypes are different.

  - There is also a new “otadata” slot, which holds the data for OTA updates. The bootloader consults this data in order to know which app to execute. If “otadata” is empty, it will execute the factory app.
    - *Note: you can create custom partition tables to suite your needs.*

Our Implementation

OTA Update Requirements

- About the Implementation
  - The User starts the OTA update by uploading a .bin file over the web page.
  - The OTA Firmware Update is performed, and the web page displays the status.
  - Once updated the web page is no longer available and the ESP32 will restart.
  - When we test the update, we'll verify the above by uploading a build file (.bin file) with a different SoftAP SSID and different color web page background.

Over the Air Updates (OTA) Using ESP-IDF, in Brief

Utilizing the ESP-IDF for OTA Updates

- Suggested Reading
  - About OTA Updates and API Reference → https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/system/ota.html
  - Partition Tables → https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-guides/partition-tables.html

Utilizing ESP-IDF for OTA Updates

- Configuration Steps and Notable ESP-IDF APIs Used
  - Receive the file from the web page → via the web server, calling httpd_req_recv.
  - Identify where the .bin file starts, then → call esp_ota_begin.
  - Write the first part of the data → call esp_ota_write.
  - Continue to receiving the file calling *httpd_req_recv* and *esp_ota_write* until all content is received.
  - Finish OTA update and validate newly written app image → call esp_ota_end.
  - Configure OTA data for a new boot partition → call esp_ota_set_boot_partition.
  - Restart the ESP32 → call esp_restart.

Let’s get started!