ESP32 OTA Integration Project

ESP32 OTA Integration Guide

Project: OTA Firmware Updates using ESP-IDF on ESP32-CAM and WROVER Modules
For: Zotbins@UC

Author: Hussain Mahuvawala

1. Project Overview

The goal of this project is to enable **Over-The-Air (OTA) firmware updates** on ESP32 microcontrollers (specifically ESP32-CAM and WROVER modules). This allows firmware updates to be sent wirelessly from a server (e.g., TuringPi) to microcontrollers, eliminating the need for physical flashing after the initial setup.

2. Environment Setup

Windows (VS Code + ESP-IDF Extension)

1. **Install ESP-IDF Tools** using the official installer:
   - Download from: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/get-started/windows-setup.html
2. **Install VS Code + ESP-IDF Extension**
   - Search for **"Espressif IDF"** in the Extensions tab and install it.
3. Open the ESP-IDF terminal from VS Code (Ctrl+Shift+P → ESP-IDF: Open Terminal).

macOS (Terminal + IDF Tools)

1. **Install Python & Git**

   ```
   brew install python git
   ```

2. **Clone ESP-IDF and run setup script**

   ```
   git clone --recursive https://github.com/espressif/esp-idf.git
   cd esp-idf
   ./install.sh esp32
   source export.sh
   ```

3. You’re now ready to use `idf.py` for building and flashing.

---

3. OTA Integration (Using `esp_https_ota()`)

OTA updates are implemented using Espressif's built-in HTTPS OTA function. This ensures firmware is securely downloaded and installed.

Key Steps:

1. Connect device to Wi-Fi
2. Create OTA task using `esp_https_ota()`
3. Provide firmware URL and TLS certificate
4. Reboot upon successful update

Sample Code Snippet:

```
esp_http_client_config_t config = {
    .url = "https://yourserver.com/firmware.bin",
    .cert_pem = (char *)server_cert_pem_start,
};
esp_https_ota(&config);
```

4. TLS Certificate

Generate a self-signed certificate:

```
openssl req -x509 -newkey rsa:2048 -keyout key.pem -out cert.pem -days
365 -nodes
cat cert.pem key.pem > server_cert.pem
```

Embed `server_cert.pem` into firmware via `CMakeLists.txt`:

```
idf_component_register(
  ...
  EMBED_TXTFILES server_cert.pem
)
```

5. Hosting the Firmware

Run a secure server to host the OTA `.bin` file:

```
python3 -m http.server 8443 --bind 0.0.0.0 --directory . --certfile
cert.pem --keyfile key.pem
```

Make sure the firmware URL in your code matches:

```
#define FIRMWARE_URL "https://<your-ip>:8443/firmware.bin"
```

6. Building and Flashing the Initial OTA Firmware

Build your project:

```
idf.py build
```

Flash the device using USB:

```
idf.py -p COMx flash monitor
```

Device will boot with OTA support enabled.

---

7. Testing OTA with TuringPi

Once you verify OTA locally:

- Host the `.bin` file on the TuringPi using HTTPS
- Ensure ESP32s can resolve the server and trust the certificate
- Trigger OTA updates from ESP32 side, and monitor via serial logs

---