Aggiornamenti OTA con Secure-Boot e Flash encryption sul dispositivo embedded ESP32

UniVR - Dipartimento di Informatica

Fabio Irimie

Tesi di laurea 2025/2026

Indice

1 Introduzione

Questo progetto consiste nell'implementazione di un sistema che permetta di aggiornare il firmware dell'ESP32 da remoto (Over The Air) tramite Wi-Fi. L'obiettivo principale è quello di attivare le funzionalità di sicurezza del microcontrollore in modo da proteggere il dispositivo da accessi non autorizzati. Le funzionalità di sicurezza includono:

- **Secure OTA**: Garantisce che il nuovo firmware sia autentico e non compromesso
- **Secure Boot**: Impedisce l'esecuzione di firmware non autorizzato
- **Flash Encryption**: Protegge i dati memorizzati nella memoria flash del dispositivo

2 Cenni teorici

2.1 Aggiornamenti OTA (Over The Air)

Gli aggiornamenti OTA permettono di aggiornare il firmware del dispositivo durante la sua normale esecuzione, senza la necessità di collegarlo fisicamente a un computer. Le modalità di aggiornamento si distinguono in base alla vulnerabilità del sistema:

- **Modalità sicura**: L'aggiornamento di alcune partizioni è resiliente, cioè garantisce l'operabilità del dispositivo anche in caso di perdita di alimentazione o di errore durante l'aggiornamento. Solo il seguente tipo di partizione supporta la modalità sicura:
  - **Application**: OTA configura la partition table in modo da avere due partizioni per l'aggiornamento (`ota_0` e `ota_1`) e una partizione per lo stato di boot (`ota_data`). Durante l'aggiornamento il nuovo firmware viene scritto nella partizione OTA attualmente non selezionata per il boot. Una volta completato l'aggiornamento, la partizione `ota_data` viene aggiornata per indicare che la partizione OTA appena scritta deve essere utilizzata al boot successivo. Se la partizione `ota_data` non contiene alcun dato il dispositivo esegue il boot dalla partizione `factory`.

    La partition table con due partizioni OTA è la seguente:

    ```
    # 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,

    ```

- **Modalità non sicura**: L'aggiornamento di alcune partizioni è vulnerabile, cioè in caso di perdita di alimentazione o di errore durante l'aggiornamento il dispositivo potrebbe non essere più operabile. Una partizione

temporanea riceve i dati della nuova immagine e, una volta completato il trasferimento, l'immagine viene copiata nella partizione di destinazione. Se l'operazione di copia viene interrotta potrebbero verificarsi problemi di boot. Le partizioni che supportano la modalità non sicura sono:

- **Bootloader**
- **Partition Table**
- **Partizioni data** (ad esempio NVS, FAT, ecc...)

2.1.1 Partizione OTA Data

Al primo avvio del dispositivo la partizione `ota_data` deve essere vuota (tutti i byte a 0xFF) in modo da far eseguire il boot dal'applicazione nella partizione `factory`. Se l'applicazione in `factory` non è presente viene eseguito il boot della prima partizione OTA disponibile (di solito `ota_0`).

Dopo il primo aggiornamento OTA, la partizione `ota_data` viene aggiornata per indicare quale partizione OTA deve essere utilizzata al successivo boot. La dimensione di `ota_data` è di due settori (0x2000 bytes = 8192 bytes) in modo da evitare errori mentre si scrive la partizione. I due settori sono cancellati indipendentemente e scritti con gli stessi dati. In questo modo se i dati dei due settori non coincidono viene usato un counter per determinare quale settore è stato scritto più recentemente.

2.1.2 App rollback

L'obiettivo dell'app rollback è quello di tenere il funzionante il dispositivo dopo un aggiornamento e permette di tornare alla versione precedente del firmware se la nuova versione non funziona correttamente (solo le partizioni OTA possono effettuare il rollback). Dopo un aggiornamento OTA con rollback attivo si hanno le seguenti possibilità:

- Se l'app funziona bene `esp_ota_mark_app_valid_cancel_rollback()` imposta lo stato dell'applicazione a `ESP_OTA_IMG_VALID`.
- Se l'app non funziona correttamente il dispositivo esegue il rollback alla versione precedente e `esp_ota_mark_app_invalid_rollback()` imposta lo stato dell'applicazione a `ESP_OTA_IMG_INVALID`.
- Se l'impostazione `CONFIG_BOOTLOADER_APP_ROLLBACK_ENABLE` è abilitata e viene effettuato un reset, allora viene effettuato un rollback senza chiamare nessuna funzione nell'applicazione. Questa opzione permette di intercettare la prima esecuzione di una nuova applicazione per confermare che funzioni correttamente.

Gli stati che controllano il processo di selezione dell'applicazione sono:

Stato	Restrizioni sulla nuova app
ESP_OTA_IMG_VALID	Nessuna restrizione. Verrà selezionata
ESP_OTA_IMG_UNDEFINED	Nessuna restrizione. Verrà selezionata
ESP_OTA_IMG_INVALID	Non verrà selezionata
ESP_OTA_IMG_ABORTED	Non verrà selezionata
ESP_OTA_IMG_NEW	Se l'opzione CONFIG_BOOTLOADER_APP_ROLLBACK_ENABLE è abilitata, l'app verrà selezionata solo una volta. Nel bootloader lo stato viene subito impostato a ESP_OTA_IMG_PENDING_VERIFY.
ESP_OTA_IMG_PENDING_VERIFY	Se l'opzione CONFIG_BOOTLOADER_APP_ROLLBACK_ENABLE è abilitata, l'app non verrà selezionata. Nel bootloader lo stato viene impostato a ESP_OTA_IMG_ABORTED.

L'impostazione di questi stati avviene nei seguenti casi:

- ESP_OTA_IMG_VALID: impostato dalla funzione esp_ota_mark_app_valid_cancel_rollback().
- ESP_OTA_IMG_UNDEFINED: impostato dalla funzione esp_ota_set_boot_partition() se l'impostazione CONFIG_BOOTLOADER_APP_ROLLBACK_ENABLE è disabilitata.
- ESP_OTA_IMG_NEW: impostato dalla funzione esp_ota_set_boot_partition() se l'impostazione CONFIG_BOOTLOADER_APP_ROLLBACK_ENABLE è abilitata.
- ESP_OTA_IMG_INVALID: impostato dalla funzione esp_ota_mark_app_invalid_rollback() o esp_ota_mark_app_invalid_rollback_and_reboot().
- ESP_OTA_IMG_ABORTED: impostato se l'operabilità dell'applicazione non è stata confermata e avviene un reboot quando l'impostazione CONFIG_BOOTLOADER_APP_ROLLBACK_ENABLE è abilitata.
- ESP_OTA_IMG_PENDING_VERIFY: impostato nel bootloader se l'impostazione CONFIG_BOOTLOADER_APP_ROLLBACK_ENABLE è abilitata e l'applicazione selezionata è nello stato ESP_OTA_IMG_NEW.

3 Implementazione

3.1 Aggiornamenti OTA

3.1.1 Configurazione del progetto

Per abilitare gli aggiornamenti OTA è stato necessario configurare il progetto ESP-IDF con le seguenti componenti:

- **Partition table e memoria flash**: è stato creato il file `sdkconfig.defaults` che contiene la configurazione di default del progetto. All'interno è stato definito l'utilizzo della partition table con due partizioni OTA:

```
CONFIG_PARTITION_TABLE_TWO_OTA=y
```

  e sono state impostate le dimensioni della memoria flash a 4MB:

```
CONFIG_ESPTOOLPY_FLASHSIZE_4MB=y
```

- **Server HTTP**: per permettere il download del nuovo firmware è stato creato un server HTTP in `server/pytest_simple_ota.py` che rende disponibile il file binario del firmware aggiornato.

  Il server richiede, nella sua cartella, la presenza di un certificato SSL per permettere connessioni sicure attraverso HTTPS. Il certificato è stato generato eseguendo il seguente comando nella cartella `server/`:

```
cd server
openssl req -x509 -newkey rsa:2048 -keyout ca_key.pem -out
    ca_cert.pem -days 365 -nodes
```

  Nota: durante la creazione del certificato nel campo `Common Name (CN)` deve essere inserito il nome host del server. Se il server viene eseguito in locale il campo deve essere impostato con l'indirizzo IP del server.

  Una volta generato il certificato bisogna flashare il file `ca_cert.pem` sul dispositivo in modo che possa verificare l'autenticità del server. Per fare ciò bisogna copiare il file nella cartella `server_certs/` del progetto:

```
cp ca_cert.pem ../server_certs
```

  Inoltre è necessario modificare il file `main/CMakeLists.txt` per includere il certificato nel firmware:

```
idf_build_get_property(project_dir PROJECT_DIR)
idf_component_register(SRCS "main.c" INCLUDE_DIRS "."
    EMBED_TXTFILES ${project_dir}/server_certs/ca_cert.pem)
```

  Una volta configurato il tutto, il server può essere avviato eseguendo il comando:

```
python pytest_simple_ota.py <BIN_DIR> <PORT> [CERT_DIR]
```

  Dove:

  - `<BIN_DIR>` è la cartella che contiene il file binario del firmware.
  - `<PORT>` è la porta su cui eseguire il server HTTP.
  - `[CERT_DIR]` (opzionale) è cartella che contiene il certificato SSL del server. Se non viene specificata viene usata la cartella corrente.

  Se tutto è andato a buon fine l'output del server sarà simile al seguente:

```
$ python pytest_simple_ota.py build 8070
Starting HTTPS server at "https://:8070"
192.168.10.106 - - [01/Jan/2026 12:00:00] "GET /
    esp32_secure_ota.bin HTTP/1.1" 200 -
```

- **Supporto del versionamento**: per tenere traccia delle versioni del firmware è stato aggiunto il file `version.txt` nella cartella principale del progetto che contiene il numero di versione corrente. Questo numero viene inserito nel file binario del firmware durante la compilazione e può essere letto dall'applicazione per confrontare la versione corrente con quella disponibile sul server ed evitare aggiornamenti non necessari.
- **Accesso Wi-Fi**: per connettere il dispositivo alla rete Wi-Fi è stata implementata la logica di connessione nel file `main/wifi.c`.
- **Configurazione Wi-Fi e server HTTP**: per fornire le credenziali di accesso alla rete Wi-Fi e l'indirizzo del server HTTP è stata creata una sezione nel menu di configurazione del progetto (menuconfig), nel file `main/Kconfig.projbuild`. Per accedere al menuconfig bisogna eseguire il comando:

```
idf.py menuconfig
```

  e navigare fino alla sezione `Over The Air Updates configuration` dove è possibile impostare:

  - **Wifi configuration**
    - `SSID`: nome della rete Wi-Fi
    - `Password`: password della rete Wi-Fi
    - `Maximum retries`: numero massimo di tentativi di riconnessione
    - `WPA3 SAE mode selection`: modalità di autenticazione WPA3
    - `Password identifier`: identificatore della password WPA3
    - `WiFi Scan auth mode threshold`: modalità di autenticazione minima accettata durante la scansione delle reti
  - **Upgrade server**
    - `Firmware upgrade URL`: URL del server HTTP che ospita il file binario del firmware. Deve essere nel formato:

      ```
      https://<host-ip-address>:<host-port>/<firmware-image-filename>
      ```

      dove:
      - `<host-ip-address>`: hostname o indirizzo IP del server HTTP.
      - `<host-port>`: porta su cui è in esecuzione il server HTTP.
      - `<firmware-image-filename>`: nome del file binario del firmware. **Deve coincidere con il nome del file messo a disposizione dal server**.
    - `Skip server certificate CN field check`: se abilitato, il dispositivo non verificherà il campo `Common Name (CN)` del certificato SSL. Utile per testare il server in locale senza dover generare un certificato con l'indirizzo IP come CN.
    - `Skip firmware version check`: se abilitato, il dispositivo non confronterà la versione del firmware corrente con quella disponibile sul server prima di effettuare l'aggiornamento OTA.
    - `OTA recieve timeout`: tempo massimo (in millisecondi) per ricevere una risposta dal server HTTP durante l'aggiornamento OTA.

3.1.2 Connessione Wi-Fi

La connessione alla rete Wi-Fi viene gestita nel file `main/wifi.c` che fornisce la funzione `connect_wifi()`. Questa funzione gestisce la connessione alla rete Wi-Fi utilizzando le credenziali fornite nel menuconfig e la riconnessione in caso di disconnessione. La funzione esegue i seguenti passaggi:

1. Inizializza il driver Wi-Fi
2. Chiama un handler per gestire gli eventi di connessione e disconnessione dalla rete Wi-Fi
3. Chiama un handler per gestire l'acquisizione dell'indirizzo IP
4. Imposta il device in modalità stazione Wi-Fi con la configurazione fornita dal menuconfig
5. Avvia la connessione alla rete Wi-Fi
6. Attende fino a quando il dispositivo non si connette alla rete o raggiunge il numero massimo di tentativi di riconnessione
7. Restituisce lo stato della connessione (successo o fallimento)

3.1.3 Aggiornamento OTA

L'aggiornamento OTA viene gestito nel file `main/ota.c` che fornisce le funzione `download_new_firmware()` e `diagnose_new_firmware()`. La funzione `download_new_firmware()` connette il dispositivo al server HTTP e scarica il nuovo firmware, se disponibile, e lo imposta come nuova applicazione di boot. I passi eseguiti sono i seguenti:

1. Recupera le partizioni di boot e OTA correnti
2. Si connette al server HTTP utilizzando l'URL fornito nel menuconfig
3. Recupera la partizione OTA non attualmente in uso per il boot
4. Legge il file binario del firmware dal server HTTP
5. Confronta la versione del firmware corrente con quella scaricata
    - Se l'aggiornamento non è necessario viene eseguito un loop infinito che attende un reset del dispositivo per ritentare l'aggiornamento
6. Inizializza l'aggiornamento OTA
7. Scrive i dati del firmware scaricato nella partizione OTA non in uso
8. Controlla che tutti i dati siano stati scritti correttamente
9. Disattiva l'aggiornamento OTA
10. Imposta la partizione OTA appena scritta come partizione di boot
11. Riavvia il dispositivo per eseguire il nuovo firmware

La funzione `diagnose_new_firmware()` permette di verificare se il nuovo firmware funziona correttamente e, in caso negativo, effettua il rollback alla versione precedente. I passi eseguiti sono i seguenti:

1. Controlla se l'applicazione corrente è al primo avvio dopo un aggiornamento OTA
2. Esegue la funzione di diagnosi del firmware (in questo caso un semplice delay di 5 secondi che simula un controllo)
    - Se la diagnosi ha esito positivo, conferma che il nuovo firmware è valido e cancella il rollback
    - Se la diagnosi ha esito negativo, marca il firmware come non valido ed esegue il rollback alla versione precedente

3.1.4 Applicazione principale

L'applicazione principale si trova nel file `main/main.c` e utilizza le funzioni definite nei file `wifi.c` e `ota.c` per connettere il dispositivo alla rete Wi-Fi e scaricare il nuovo firmware. I passi eseguiti sono i seguenti:

1. Controlla il nuovo firmware tramite la funzione `diagnose_new_firmware()`
2. Si connette alla rete Wi-Fi tramite la funzione `connect_wifi()`
3. Crea una task per gestire l'aggiornamento OTA chiamando la funzione `download_new_firmware()`
4. Crea una task per eseguire l'applicazione principale (in questo caso un semplice loop che stampa un messaggio ogni 5 secondi)

Bibliografia

[1] Espressif Systems. *Over The Air Examples*. URL: `https://github.com/espressif/esp-idf/tree/master/examples/system/ota`. (accessed: 20.10.2025).

[2] Espressif Systems. *Over The Air Updates (OTA)*. URL: `https://docs.espressif.com/projects/esp-idf/en/stable/esp32/api-reference/system/ota.html`. (accessed: 20.10.2025).