УДК 004.31

Н. О. Черневський

АРХІТЕКТУРНІ РІШЕННЯ ТА МЕХАНІЗМИ БЕЗДРОТОВОГО ОНОВЛЕННЯ ПРОШИВОК МІКРОКОНТРОЛЕРІВ

Вінницький Національний Технічний Університет

Анотація

У роботі розглянуто архітектуру систем FOTA (Firmware Over-The-Air) оновлення для мікроконтролерів в IoT-системах. Проаналізовано практичні проблеми: розміщення прошивок в обмеженій пам'яті, верифікацію цілісності та автентичності оновлень, механізми відкату при помилці. Описано реалізацію на платформах ESP32 та STM32, розглянуто методи забезпечення безпеки.

Ключові слова: *FOTA, прошивка, мікроконтролер, OTA, ESP32, STM32, Dual Boot.*

Abstract

The paper examines the architecture of FOTA (Firmware Over-The-Air) update systems for microcontrollers in IoT applications. Practical challenges are analyzed: firmware placement in limited memory, integrity and authenticity verification, and rollback mechanisms upon failure. Implementation on ESP32 and STM32 platforms is discussed, along with security methods.

Keywords: FOTA, firmware, microcontroller, OTA, ESP32, STM32, Dual Boot.

Вступ

Розгортання IoT-пристроїв у реальних умовах це лише перший етап життєвого циклу системи. Справжні виклики виникають, коли з'являється потреба в їх оновленні. Якщо сенсор встановлено на даху багатоповерхівки або на віддаленому промисловому об'єкті, його фізичне обслуговування для прошивки через USB є економічно недоцільним. Тому використання бездротового оновлення є дуже зручним інструментом.

При прошивці методом OTA важливо, щоб нова прошивка мала коректний розмір, не пошкоджувала існуючі конфігураційні дані, успішно запускалася, а у разі збою пристрій міг автоматично повернутися до попередньої стабільної версії. Крім того, виникають жорсткі вимоги до безпеки. Завантажувати неперевірений код у мікроконтролер, що керує критичною інфраструктурою, вкрай небезпечно.

Архітектура систем віддаленого оновлення

Базова архітектура FOTA спирається на розподіл Flash-пам'яті на логічні розділи. У мікроконтролерах ESP32 це реалізовано так: у базовій частині пам'яті знаходиться незмінний bootloader (завантажувач), а інший простір поділено на слоти для активної прошивки та оновлень (OTA-розділи). Нова прошивка завантажується не поверх робочої, а в резервний розділ. Bootloader перевіряє цілісність нового образу, і якщо перевірка успішна, перемикає активний розділ на новий.

Сучасні мікроконтролери STM32 (наприклад, серій G4, L4, H7) використовують апаратний механізм Dual Bank Flash. Активна прошивка розташовується в першому банку пам'яті, а нова завантажується у другий. Після успішного завантаження bootloader змінює спеціальний біт в Option Bytes, і мікроконтролер на апаратному рівні міняє банки пам'яті місцями.

Складніше працювати з ARM-платформами без вбудованої підтримки Dual Boot, де завантажувач доводиться розробляти самостійно. У таких випадках доцільно використовувати готові рішення на кшталт MCUboot, які керують слотами пам'яті та забезпечують безпечний механізм відкату (rollback).

Практичні виклики під час OTA оновлень

Перша проблема це обмежений обсяг Flash-пам'яті. Наприклад, у ESP32 із 4 МБ пам'яті розмістити одночасно поточну прошивку (яка може займати 1–2 МБ), файлову систему та OTA-розділ буває складно. Вирішенням є перехід на чип із більшим обсягом пам'яті або застосування алгоритмів стиснення (наприклад, LZMA чи LZ4). У цьому разі прошивка передається у стиснутому вигляді, а

bootloader розпаковує її на льоту під час запису з резервного слота в основний робочий розділ Flash-пам'яті.

Друга проблема це надійність мережевої передачі. Завантаження 2 МБ даних через нестабільне Wi-Fi або стільникове з'єднання може тривати довго, і ймовірність розірвання зв'язку є високою. Надійна реалізація передбачає передачу прошивки невеликими блоками (по 1–4 КБ) із фіксацією індексу останнього успішно прийнятого блоку. У разі переривання з'єднання процес поновлюється з місця зупинки (дозавантаження), а не починається спочатку.

Третя проблема це перевірка цілісності. Використання лише алгоритму CRC32 є недостатнім для критичних систем. У разі виникнення специфічних помилок під час передачі, які не виявляються CRC, мікроконтролер може спробувати запустити пошкоджений код. Більш надійним підходом є верифікація за допомогою HMAC-SHA256. Сервер передає прошивку разом із криптографічним хешем, який bootloader перевіряє перед її активацією.

Безпека процесу оновлення

Цифровий підпис прошивки це базовий рівень захисту від підміни. На стороні сервера образ підписується приватним ключем розробника, а на пристрої (у bootloader) зберігається відповідний публічний ключ. Якщо зловмисник перехопить трафік і змінить хоча б один байт, перевірка підпису завершиться помилкою, і оновлення буде відхилено.

Жорстке версіонування прошивок запобігає атакам типу downgrade. Якщо зловмисник спробує оновити пристрій на стару версію ПЗ, що містить відомі вразливості, завантажувач заблокує цю дію. Новий образ приймається лише за умови, що його номер версії (або security counter) є вищим за поточний.

Фізична ізоляція завантажувача гарантує, що процес FOTA випадково не зашкодить самому механізму оновлення. Bootloader записується в захищений сектор пам'яті виключно через JTAG або спеціальний режим DFU (Device Firmware Upgrade). Мережеві оновлення записуються лише в призначені для цього слоти (App partitions).

Механізм відкату (rollback) при помилці гарантує повернення пристрою до попередньої працездатної версії, якщо нова прошивка містить критичний баг і викликає bootloop (циклічне перезавантаження). Це реалізується за допомогою лічильника перезавантажень та сторожового таймера (Watchdog). Якщо після оновлення система не змогла стабільно пропрацювати заданий час і підтвердити свою життєздатність, завантажувач автоматично позначає нову прошивку як недійсну і перемикається на попередню робочу.

Висновки

Технологія FOTA є невід'ємною складовою сучасних IoT-систем, без якої керування великою кількістю автономних пристроїв стає практично неможливим. На практиці створення системи бездротового оновлення зводиться до чотирьох основних завдань: оптимізації Flash-пам'яті, забезпечення безперебійної передачі даних, перевірки цілісності прошивки та захисту від її підміни.

Платформи ESP32 та STM32 пропонують потужні апаратні та програмні механізми для реалізації FOTA (OTA-розділи, Dual Bank Flash), що дозволяє інтегрувати безпечне оновлення без необхідності створення завантажувача з нуля. Однак готові інструменти не пробачають архітектурних помилок. Нехтування базовими принципами розробки може призвести до безповоротної втрати контролю над пристроями.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. ESP-IDF OTA Updates Documentation. URL: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/system/ota.html (дата звернення: 01.06.2026).

2. Firmware Over-The-Air (FOTA) Updates on ESP32 URL: https://www.hackster.io/cieslam/firmware-over-the-air-fota-updates-on-esp32-4a2d75 (дата звернення: 01.06.2026).

3. STM32 Application Note AN4657: In-application programming. URL: https://www.st.com/resource/en/application_note/an4657-in-application-programming-on-stm32l-microcontrollers-stmicroelectronics.pdf (дата звернення: 01.06.2026).

4. ESP HTTPS OTA - ESP32 URL: https://docs.espressif.com/projects/esp-idf/en/stable/esp32/api-reference/system/esp_https_ota.html (дата звернення: 27.05.2026).

Черневський Назар Олександрович — студент групи 2КІ-25м, факультет інформаційних технологій та комп'ютерної інженерії, Вінницький Національний Технічний Університет, Вінниця, e-mail: chernevskijnazar@gmail.com

Chernevskyi Nazar Oleksandrovich — student of group 2KI-25m, faculty of information technologies and computer engineering, Vinnytsia National Technical University, Vinnytsia, e-mail: chernevskijnazar@gmail.com