УДК 004.312

Н. О. Черневський

ОПТИМІЗАЦІЯ ЕНЕРГОСПОЖИВАННЯ МІКРОКОНТРОЛЕРІВ В IOT-ПРИСТРОЯХ НА ОСНОВІ РЕЖИМІВ ГЛИБОКОГО СНУ

Вінницький національний технічний університет

Анотація

У роботі розглянуто підходи до мінімізації енергоспоживання мікроконтролерів в автономних IoT-пристроях із застосуванням режимів глибокого сну. Проаналізовано апаратні механізми deep sleep на платформах ESP32 та STM32, досліджено вплив частоти пробуджень, джерел переривань та периферії на загальне споживання струму. Наведено практичні рекомендації щодо вибору режиму сну залежно від сценарію роботи пристрою.

Ключові слова: IoT, deep sleep, мікроконтролер, ESP32, STM32, енергоефективність, автономне живлення.

Abstract

The paper examines approaches to minimizing power consumption in autonomous IoT devices using deep sleep modes. Hardware sleep mechanisms on ESP32 and STM32 platforms are analyzed, along with the impact of wake-up frequency, interrupt sources, and peripheral state on total current draw. Practical guidelines are provided for selecting the appropriate sleep mode based on the device's operating scenario.

Keywords: IoT, deep sleep, microcontroller, ESP32, STM32, energy efficiency, battery-powered devices.

Вступ

Більшість IoT-пристроїв живляться від батарей або невеликих акумуляторів. Для таких пристроїв термін служби батареї часто важливіший за обчислювальну потужність. Мікроконтролер у активному режимі споживає від 30 до 300 мА, тоді як у режимі глибокого сну цей показник падає до одиниць мікроампер. Різниця становить чотири-п'ять порядків.

Проте на практиці реалізація енергоефективних систем стикається з певними труднощами. Периферія після пробудження може не ініціалізуватись коректно, вміст RAM зникає, деякі протоколи зв'язку взагалі погано поєднуються з циклами сну. Також може бути проблема витоку струму через незакриті GPIO, пристрій нібито спить, а батарея все одно сідає. У цій роботі розглянуто основні підходи до організації режимів сну і те, де найчастіше виникають помилки.

Режими енергозбереження та методи пробудження

Сучасні мікроконтролери підтримують кілька рівнів енергозбереження. На прикладі ESP32 це п'ять режимів: активний, modem sleep, light sleep, deep sleep та hibernation. Ключова відмінність між ними полягає в блоках, які залишаються живленими. У light sleep зберігається вміст RAM та стан периферії, процесор зупиняється, пробудження відбувається за мілісекунди. У deep sleep живиться лише сопроцесор ULP та RTC-пам'ять об'ємом 8 КБ, споживання становить близько 10–150 мкА залежно від конфігурації. Режим hibernation вимикає навіть ULP і залишає лише RTC-таймер, знижуючи струм до 5 мкА, але втрачаючи всі джерела пробудження окрім таймера та зовнішнього сигналу EXT0.

На платформі STM32 є аналогічна ієрархія: sleep, stop та standby. Stop-режим зберігає вміст RAM і регістри периферії, споживає 1–10 мкА, пробуджується від EXTI, UART або таймера. Standby вимикає все окрім RTC та резервного домену, знижуючи струм до 0,5–2 мкА, але вимагає повної реініціалізації після пробудження, аналогічно до скидання.

Вибір джерела пробудження безпосередньо визначає загальне енергоспоживання пристрою. Найпростіший варіант це пробудження за таймером RTC. Якщо датчик має зчитувати покази кожні 5 хвилин, пристрій проводить в активному режимі близько 300 мс, решту часу у deep sleep. При споживанні в активному режимі 80 мА та 150 мкА у сні середній струм складе приблизно 0,23 мА, що дає близько 434 дня роботи від батареї ємністю 2400 мАг.

Пробудження від зовнішнього переривання (EXTI на STM32, EXT0/EXT1 на ESP32) дозволяє реагувати на події без постійного опитування. Це зручно для охоронних датчиків, кнопок або сигналів готовності від зовнішніх мікросхем. Пробудження від touch-сенсора або аналогового порогового компаратора дає ще більшу гнучкість, але вимагає точної апаратної реалізації, щоб уникнути хибних спрацювань від шумів.

Типові помилки та джерела витоку струму

На практиці значення споживання, описане в технічній документації, часто не досягається через кілька поширених причин.

По-перше, якщо вивід залишено у стані High (логічна одиниця), і він продовжує живити зовнішнє навантаження. Також, якщо вивід налаштований як вхід без внутрішньої або зовнішньої підтяжки (залишається плаваючим), він ловить електромагнітні наведення, через що вхідний CMOS-буфер починає хаотично перемикатись і споживати певний струм.

По-друге через активні периферійні мікросхеми. Сенсори з постійним живленням (наприклад, BME280 у normal mode або GPS-модуль) продовжують споживати струм незалежно від стану мікроконтролера. Рішенням є програмне переведення їх у режим сну через відповідний регістр або повне зняття з них живлення за допомогою P-канального MOSFET, керованого від GPIO.

По-третє через неоптимальне пробудження. Якщо після пробудження пристрій підключається до Wi-Fi для надсилання одного пакету, час встановлення з'єднання (зазвичай 2–5 секунд) домінує у загальному балансі енергії. Протоколи ESP-NOW або BLE з пакетами advertisement усувають цю затримку, дозволяючи передати дані за 10–50 мс.

Висновки

Застосування режимів deep sleep дозволяє збільшити термін роботи IoT-пристрою від батареї в десятки і сотні разів порівняно з постійно активним режимом. При цьому вибір конкретного режиму залежить від трьох параметрів: необхідного часу реакції на подію, обсягу стану, який потрібно зберегти між пробудженнями, та набору доступних джерел переривань.

Реальне споживання пристрою у сні визначається не лише режимом мікроконтролера, а й станом усієї зовнішньої схеми. Закриття GPIO, відключення периферії та вибір швидкого протоколу передачі замість Wi-Fi на практиці дають більший ефект, ніж перехід між суміжними режимами сну самого мікроконтролера.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. ESP32 Technical Reference Manual. Sleep Modes. URL: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/system/sleep_modes.html (дата звернення: 24.05.2026).

2. STM32 Application Note AN4621: Low-power modes. URL: https://www.st.com/resource/en/application_note/an4621-stm32l4-and-stm32l4plus-ultralowpower-features-overview-stmicroelectronics.pdf (дата звернення: 24.05.2026).

3. Texas Instruments: MSP430 Ultra-Low-Power Design Guide. URL: https://www.ti.com/lit/ug/slau208/slau208.pdf (дата звернення: 24.05.2026).

4. Espressif ESP-NOW documentation. URL:https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/network/esp_now.html (дата звернення: 24.05.2026).

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