УДК 004.312.46:004.7

ОЦІНКА ЕФЕКТИВНОСТІ MICROPYTHON ТА C/C++ ДЛЯ ВБУДОВАНИХ СИСТЕМ НА БАЗІ МІКРОКОНТРОЛЕРА ESP32

Оніщенко А.В.
Науковий керівник – к.т.н., асистент Волох А.В.
e-mail: andrii.onishchenko@nure.ua
Харківський національний університет радіоелектроніки, каф. МІРЕС,
студ. наук. гурток «Розробка вбудованих систем»,
м. Харків, Україна

This article presents a performance comparison between C/C++ and MicroPython on ESP32 microcontrollers, using the implementation of CRC-32 and Fast Fourier Transform (FFT) algorithms as examples. While C/C++ is the native programming language for the ESP32, alternative languages such as MicroPython are gaining increasing popularity among developers. The implementation and testing of these two data and signal processing algorithms on the ESP32 microcontroller demonstrated significantly higher performance when using C. At the same time, MicroPython proves to be more suitable for tasks with lower performance requirements, where ease of use and rapid development are prioritized.

У вбудованих системах та пристроях Інтернету речей (IoT) використовується широкий спектр різноманітних мікроконтролерів для вирішення задач збору, обробки та передачі інформації. Перевагами використання мікроконтролерів (МК) ESP32 для IoT є їх енергоефективність, наявність вбудованих бездротових інтерфейсів, можливість використання готових бібліотек та різноманітних мов при програмуванні цих МК [1]. Хоча основною мовою програмування МК ESP32 є мова системного програмування C/C++, все більшу популярність серед розробників здобуває використання альтернативних мов, таких як Rust, TinyGo, MicroPython. В зв’язку з цим представляє інтерес порівняння ефективності роботи МК ESP32 з точки зору швидкодії програм, написаних на C/C++, MicroPython, на прикладі реалізації найбільш поширених на практиці алгоритмів обробки даних та сигналів.

Порівняння продуктивності використання мов програмування C та MicroPython в МК ESP32 виконувалося з використанням реалізації на двох мовах алгоритму обчислення циклічного надлишкового коду CRC32 та швидкого перетворення Фур’є (ШПФ). Ці алгоритми були обрані завдяки: популярності їх використання в практичних розробках на МК; присутності в існуючих бенчмарках, таких як MiBench [2], орієнтованих на тестування вбудовуваних процесів; наявності готових протестованих та оптимізованих реалізацій алгоритмів мовою C в бібліотеках з відкритим вихідним кодом з компонентів ESP-IDF (фреймворку від виробника Espressif) [3]. Використ-

тання існуючих реалізацій з відкритим вихідним кодом спростило портування алгоритмів на мову MicroPython, а також перевірку правильності їх роботи після перенесення. Реалізація вказаних алгоритмів на MicroPython не використовувала наявні готові рішення з бібліотек на Python 3, щоб запобігти виклику заздалегідь скомпільованих функції на мові C. Тестування проводилося з використанням плати розробника ESP32-WROOM-32 на частоті процесора 160 МГц, одній зі стандартних робочих частот МК в багатьох прикладних задачах.

На рисунку 1 представлено результати середніх значень часу виконання алгоритму CRC32, реалізованого на мовах C та MicroPython. Усереднення значень часу проводилося на основі 100 ітерацій для різної кількості вхідних даних 0, 16, 32, 64, 128, 256, 512 и 1024 байт. Шкала вимірювання значень часу наведена в логарифмічному масштабі.

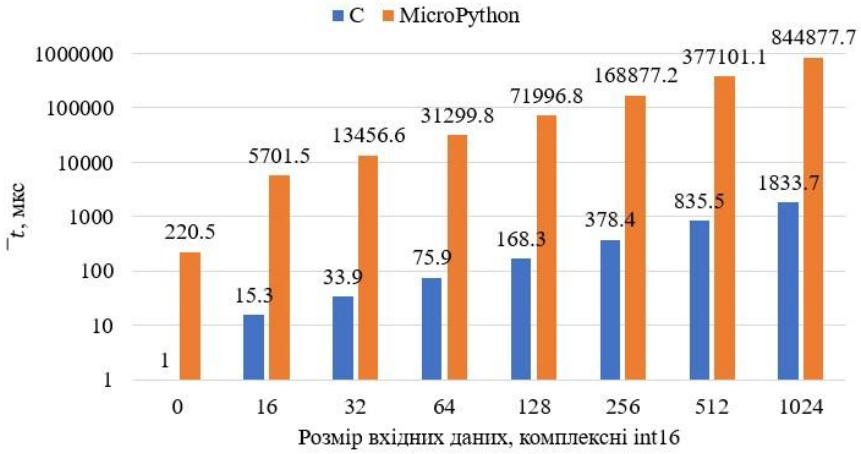

Рисунок 1 – Середній час виконання алгоритму CRC32

На рисунку 2 представлено результати середніх значень часу виконання алгоритму ШПФ. Тестування проводилося для тої ж кількості ітерацій та розмірів вхідних даних, що і в першому випадку. Але вхідні дані для ШПФ представляли собою комплексні числа в форматі int16, що дозволило краще протестувати архітектурну ефективність мов: як вони працюють з пам'яттю, бітовими зсувами та великими масивами цілих чисел.

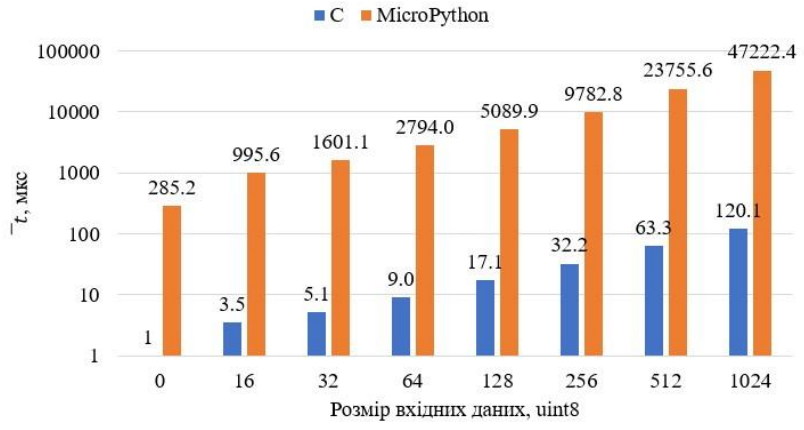

Рисунок 2 – Середній час виконання алгоритму ШПФ

За результатами тестування роботи МК ESP32 для двох обраних алгоритмів обробки даних та сигналів, очікувано MicroPython показав продуктивність нижчу в десятки або сотні разів, порівняно з реалізацією алгоритмів мовою С. Причинами такої низької продуктивності MicroPython, при використанні в МК, можна назвати наступні чинники: додаткові затримки на інтерпретацію байт-коду в реальному часі; відсутністю додаткової низькорівневої оптимізації коду на етапі компіляції з урахуванням специфіки апаратної платформи; призупиненням роботи програми на час видалення непотрібних об'єктів збирачем сміття; наявністю додаткових накладних витрати на перевірку типів та виконання операцій з об'єктами при роботі з масивами та простими типами даних в Python. Використання C/C++ в МК ESP32 є безальтернативним для систем з жорсткими часовими обмеженнями (Real-Time системи), коли критично важливим фактором є час виконання та реакції програми. Тоді як мова MicroPython краще підходить для програмування МК, коли необхідна швидка та більш проста розробка з меншими вимогами до швидкодії системи, а також як альтернатива складному синтаксису C/C++ для задач швидкого прототипування та при використанні в учбових цілях.

Список використаних джерел:

1. Maier A., Sharp A., Vagapov Y. Comparative Analysis and Practical Implementation of the ESP32 Microcontroller Module for the Internet of Things // In Proceedings of the 2017 Internet Technologies and Applications (ITA). 2017. P. 143–148.

2. Guthaus M.R., Ringenberg J.S., Ernst D. and etc. A Free, Commercially Representative Embedded Benchmark Suite // In Proceedings of the 4th Annual IEEE International Workshop on Workload Characterization. 2001. P. 3–14.

3. Espressif DSP Library. URL: https://docs.espressif.com/projects/esp-dsp/en/latest/esp-dsp-library.html (дата звернення: 27.02.2026).