PHÂN TÍCH HIỆU NĂNG ĐA XỬ LÝ ĐỐI XỨNG (SMP) TRÊN VI ĐIỀU KHIỂN ESP32-WROOM-32E-N4

Quách Thị Hạnh[1*], Nguyễn Thị Tố Uyên[1]
[*]Tác giả liên hệ, email: hanhqt.dttt@hou.edu.vn. ORCID: 0009-0007-9968-4543

Ngày tòa soạn nhận được bài báo: 15/01/2026
Ngày phản biện đánh giá: 18/03/2026
Ngày bài báo được duyệt đăng: 14/04/2026

DOI: 10.59266/houjs.2026.1179

Tóm tắt: *Nghiên cứu này tập trung đánh giá hiệu quả thực tế của kiến trúc đa xử lý đối xứng (Symmetrical Multiprocessing - SMP) trên dòng vi điều khiển ESP32-WROOM-32E-N4. Bằng phương pháp thực nghiệm, chúng tôi so sánh thời gian thực thi của các thuật toán tính toán dấu phẩy động (floating-point arithmetic) trong hai kịch bản: chạy tuần tự trên đơn nhân và chạy song song trên hai nhân. Kết quả thực nghiệm cho thấy, đối với các tác vụ tính toán chuyên sâu (compute-bound) có khối lượng đủ lớn, việc tận dụng lõi thứ hai giúp hệ thống đạt mức tăng tốc 1.995 lần, tương đương hiệu suất 99.7% so với lý thuyết. Tuy nhiên, nghiên cứu cũng chỉ ra rằng ở các tác vụ có thời gian thực thi ngắn (dưới 4 mili giây), chi phí khởi tạo tác vụ và quản lý ngữ cảnh của hệ điều hành thời gian thực (FreeRTOS) sẽ làm giảm đáng kể hiệu quả song song hóa. Những phát hiện này cung cấp cơ sở quan trọng cho việc tối ưu hóa ứng dụng điện toán biên (Edge Computing) trên các thiết bị IoT giá thành thấp.*

Từ khóa: *đa xử lý đối xứng (Symmetrical Multiprocessing - SMP), ESP32, FreeRTOS, kiểm chuẩn hiệu năng (Performance Benchmarking), vi điều khiển lõi kép (Dual-core Microcontroller)*

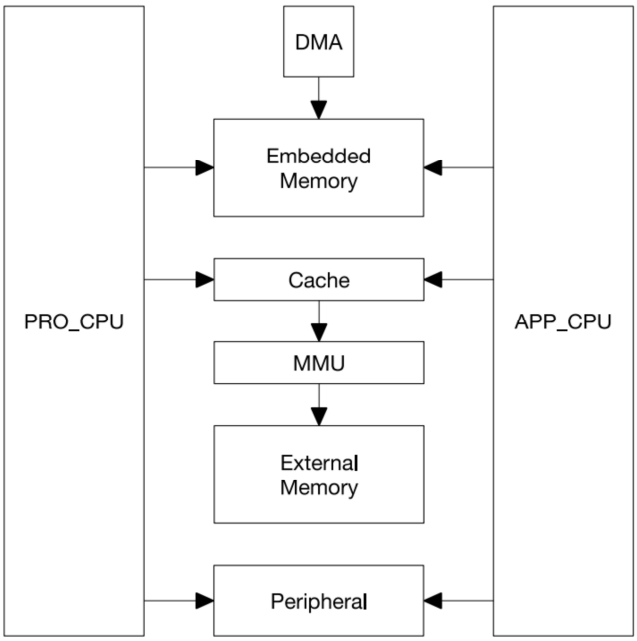
I. Đặt vấn đề

Sự bùng nổ của Internet vạn vật (IoT) trong thập kỷ qua đã đặt ra những yêu cầu ngày càng khắt khe về năng lực xử lý tại biên (Edge). Các thiết bị nhúng hiện đại không chỉ đơn thuần thu thập dữ liệu từ cảm biến mà còn phải thực hiện các tác vụ phức tạp như mã hóa, xử lý tín hiệu số (DSP) và chạy các mô hình học máy cơ bản (Shi và cộng sự, 2016). Để đáp ứng nhu cầu này mà vẫn đảm bảo hiệu quả năng lượng, các nhà sản xuất bán dẫn đã chuyển dịch từ kiến trúc đơn nhân truyền thống sang kiến trúc đa nhân. Trong phân khúc vi điều khiển giá rẻ, dòng chip ESP32 của Espressif Systems là một đại diện tiêu biểu cho xu hướng này với kiến trúc lõi kép (Dual-Core).

Đối tượng nghiên cứu chính của bài báo cáo này là module ESP32-WROOM-32E-N4, sử dụng vi xử lý Xtensa® 32-bit LX6 lõi kép. Theo tài liệu kỹ thuật từ nhà

[1] Khoa Điện - Điện tử, Trường Đại học Mở Hà Nội, Hà Nội, Việt Nam

sản xuất, vi xử lý này có khả năng vận hành ở tần số xung nhịp lên đến 240 MHz và sở hữu bộ nhớ SRAM 520 KB (*ESP32-WROOM-32E & ESP32-WROOM-32UE Datasheet*, n.d.). Một đặc điểm quan trọng của phiên bản silicon V3 trên module này là sự cải thiện về độ ổn định so với các thế hệ trước, cho khả năng khai thác tối đa sức mạnh của cả hai lõi trong môi trường đa xử lý đối xứng.



Hình 1. Cấu trúc hệ thống lõi kép PRO_CPU và APP_CPU trên Xtensa LX6

Mặc dù kiến trúc phần cứng hỗ trợ hai lõi vật lý, việc đạt được hiệu suất gấp đôi trong thực tế là một thách thức không nhỏ. Theo Định luật Amdahl, tốc độ tăng tốc tối đa của một chương trình song song luôn bị giới hạn bởi thành phần tuần tự của chương trình đó (Amdahl, 1967). Trong môi trường vi điều khiển sử dụng hệ điều hành thời gian thực FreeRTOS, thành phần tuần tự này bao gồm chi phí khởi tạo tác vụ (task creation overhead), chi phí chuyển đổi ngữ cảnh (context switching) và độ trễ do tranh chấp tài nguyên bộ nhớ chung (Barry, 2024). Nếu không được quản lý tốt, các yếu tố này có thể triệt tiêu lợi ích của việc xử lý song song, thậm chí khiến hệ thống đa nhân hoạt động chậm hơn đơn nhân trong một số trường hợp cụ thể.

Bài báo cáo này trình bày phương pháp đo đạc và phân tích định lượng hiệu năng của ESP32 thông qua một loạt các bài kiểm thử (benchmark) với độ phức tạp tăng dần. Chúng tôi thiết kế một thuật toán tập trung vào tính toán số học trên thanh ghi để giảm thiểu sự phụ thuộc vào bộ nhớ Flash, từ đó cô lập và đo lường chính xác năng lực xử lý của CPU. Kết quả nghiên cứu sẽ làm rõ mối tương quan giữa kích thước tác vụ và hiệu suất song song, đồng thời đề xuất các chiến lược lập trình tối ưu để tránh các lỗi phổ biến như treo hệ thống do Watchdog Timer hay xung đột bộ nhớ đệm (cache contention).

II. Phương pháp nghiên cứu

2.1. Thiết lập phần cứng và môi trường phát triển

Đối tượng thí nghiệm là module ESP32-WROOM-32E-N4 (sử dụng silicon revision V3), được trang bị vi xử lý Xtensa® Dual-Core 32-bit LX6 vận

hành ở tần số xung nhịp cố định 240 MHz (*ESP32-WROOM-32E & ESP32-WROOM-32UE Datasheet*, n.d.). Việc lựa chọn phiên bản silicon V3 nhằm loại bỏ các lỗi phần cứng (errata) tồn tại trên các phiên bản tiền nhiệm, đảm bảo độ ổn định tối đa khi cả hai nhân hoạt động ở công suất cực đại.

Về môi trường phần mềm, thay vì sử dụng Arduino IDE truyền thống, chúng tôi phát triển mã nguồn trên nền tảng Visual Studio Code tích hợp PlatformIO. Đây là hệ thống quản lý dự án chuyên nghiệp cho phép kiểm soát chặt chẽ các phụ thuộc, thư viện và thông số biên dịch. Cấu hình dự án được thiết lập trong tệp platformio. ini với framework arduino, tốc độ giám sát (monitor speed) 115200 baud và tần số CPU được khóa cứng ở mức 240 MHz để loại bỏ các biến động do cơ chế tiết kiệm năng lượng (Dynamic Frequency Scaling) gây ra.

2.2. Thiết kế thuật toán kiểm chuẩn

Để đánh giá khách quan hiệu năng xử lý của kiến trúc đa nhân Xtensa® LX6, yêu cầu tiên quyết là phải cô lập được năng lực tính toán của CPU khỏi các yếu tố ngoại vi khác như tốc độ truy xuất bộ nhớ hay độ trễ của các ngoại vi nhập/xuất (I/O). Do đó, thuật toán kiểm chuẩn trong nghiên cứu này được thiết kế theo định hướng register-based compute-bound, tức giới hạn của tốc độ tính toán dựa hoàn toàn vào các thanh ghi (registers).

Cụ thể, chúng tôi xây dựng một hàm tính toán thực hiện lặp đi lặp lại các phép toán số học dấu phẩy động phức tạp, bao gồm phép căn bậc hai, phép nhân và phép chia số thực. Khác với các thuật toán phụ thuộc bộ nhớ (memory-bound) (Williams và cộng sự, 2009) vốn đòi hỏi băng thông lớn để luân chuyển dữ liệu giữa RAM và CPU, thuật toán này chủ yếu thao tác trên các thanh ghi nội bộ của bộ vi xử lý.

Đặc biệt, để đảm bảo tính chính xác của phép đo trong môi trường biên dịch tối ưu (như cờ -O2 hoặc -Os thường dùng trong PlatformIO), tất cả các biến số tham gia vào vòng lặp tính toán đều được khai báo với từ khóa volatile. Kỹ thuật này chỉ thị cho trình biên dịch không được phép thực hiện các tối ưu hóa tự động (như tính toán trước kết quả hằng số hoặc loại bỏ các vòng lặp không tạo ra thay đổi trạng thái bên ngoài). Nhờ đó, chúng tôi buộc đơn vị số học dấu phẩy động (FPU) của từng nhân phải thực thi trọn vẹn từng chỉ lệnh trong thời gian thực, đảm bảo số liệu thu được phản ánh trung thực sức mạnh xử lý thô (raw performance) của vi điều khiển.

2.3. Giải pháp ổn định hệ thống và Watchdog Timer

Trong môi trường hệ điều hành thời gian thực FreeRTOS, cơ chế bảo vệ Task Watchdog Timer (TWDT) được thiết kế để giám sát và khởi động lại hệ thống nếu một tác vụ chiếm dụng CPU quá lâu mà không nhường quyền điều khiển (yield). Với các bài kiểm thử cường độ cao (lên đến 10 triệu phép tính trong thí nghiệm của chúng tôi), việc đơn nhân hoặc đa nhân chiếm dụng 100% tài nguyên CPU sẽ kích hoạt cơ chế này, gây ra hiện tượng reset hệ thống không mong muốn (*Watchdogs - ESP32 - - ESP-IDF Programming Guide v5.5.2 Documentation*, n.d.).

Để giải quyết vấn đề trên mà không làm gián đoạn luồng tính toán, chúng tôi đã can thiệp vào cấu hình của TWDT thông qua thư viện esp_task_wdt.h. Thời gian chờ (timeout) của Watchdog được mở rộng từ 5 giây mặc định lên 60 giây, đồng thời chế độ «Panic» (tự động reset) bị vô hiệu hóa. Cấu hình này cho phép các tác vụ kiểm chuẩn hoàn thành chu trình

tính toán dài mà không bị hệ điều hành ngắt quãng, trong khi vẫn duy trì cơ chế bảo vệ nền tảng ở mức độ tối thiểu.

2.4. Chiến lược phân luồng và quy trình đo lường

Mô hình thực thi song song được áp dụng là mô hình "Fork-Join". Quá trình thí nghiệm diễn ra theo trình tự sau: Trước tiên, hệ thống thực hiện đo lường thời gian xử lý tuần tự trên một nhân đơn lẻ để thiết lập mốc so sánh cơ sở (baseline). Sau khoảng thời gian nghỉ 2 giây để tản nhiệt và ổn định điện áp, hệ thống tiến hành phân chia khối lượng công việc thành hai luồng độc lập.

Chúng tôi sử dụng hàm xTaskCreatePinnedToCore của FreeRTOS để gán tác vụ con thứ nhất vào Core 0 và tác vụ con thứ hai vào Core 1. Cả hai tác vụ này được thiết lập độ ưu tiên (priority) mức 2, cao hơn mức mặc định của vòng lặp hệ thống, nhằm đảm bảo quyền ưu tiên sử dụng CPU. Thời gian thực thi được đo lường chính xác đến từng micro giây (µs) bằng hàm micros(), bắt đầu từ khi các tác vụ được khởi tạo cho đến khi tín hiệu hoàn thành được thu nhận từ cả hai luồng.

III. Kết quả và thảo luận

Quá trình kiểm chuẩn được thực hiện tự động thông qua chuỗi bài thử nghiệm với khối lượng tính toán tăng dần theo lũy thừa của 10, từ mức thấp nhất là 1.000 vòng lặp đến mức cao nhất là 10.000.000 vòng lặp. Tại mỗi mốc kiểm thử, hệ thống ghi nhận thời gian thực thi của chế độ đơn nhân (Single-Core) và chế độ đa nhân (Dual-Core) với độ chính xác tính bằng micro giây (µs) sau đó quy đổi sang mili giây (ms) để thuận tiện cho việc so sánh và trình bày.

3.1. Dữ liệu đo lường

Kết quả đo lường chi tiết được tổng hợp trong Bảng 1. Dữ liệu cho thấy thời gian xử lý tỷ lệ thuận với số lượng phép tính, phản ánh tính ổn định của vi xử lý Xtensa® LX6 khi hoạt động ở xung nhịp 240 MHz. Một chỉ số quan trọng được tính toán là Hệ số tăng tốc S (Speedup Factor), được xác định bằng tỷ lệ giữa thời gian chạy đơn nhân (T_s) chia cho thời gian chạy đa nhân (T_d):

$$S = \frac{T_s}{T_d} \quad (1)$$

Một chỉ số quan trọng khác được tính toán trong nghiên cứu là Hiệu suất Song song (Parallel Efficiency), ký hiệu là *E*. Chỉ số này đánh giá mức độ hiệu quả trong việc sử dụng tài nguyên phần cứng, cho biết hệ thống thực tế đang hoạt động bao nhiêu phần trăm so với công suất lý thuyết của cả hai nhân. Công thức tính như sau:

$$E = \frac{S}{P} * 100\% \quad (2)$$

Trong đó:

S là Hệ số tăng tốc.

P là số lượng nhân xử lý (trong thí nghiệm này, P=2).

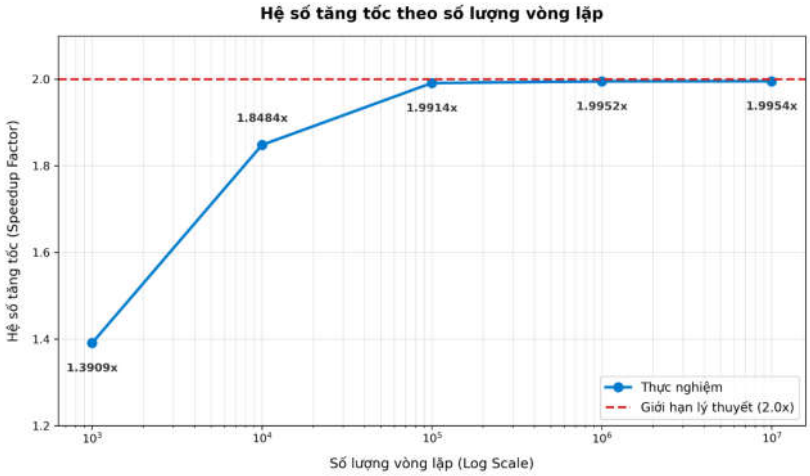
Số vòng lặp (N)	Thời gian Đơn nhân (Ts)	Thời gian Đa nhân (Td)	Hệ số tăng tốc (S)	Hiệu suất Song song (E)
1,000	4.14 ms	2.97 ms	**1.3909x**	69.5%
10,000	40.63 ms	21.98 ms	**1.8484x**	92.5%
100,000	406.22 ms	203.98 ms	**1.9914x**	99.5%
1,000,000	4,068.19 ms	2,038.98 ms	**1.9952x**	99.7%
10,000,000	40,760.67 ms	20,426.98 ms	**1.9954x**	99.7%

Bảng 1. Thời gian thực thi và Hệ số tăng tốc của ESP32-WROOM-32E-N4 qua các mức tải khác nhau

Lưu ý: Thời gian đo được đã được làm tròn đến hai chữ số thập phân sau khi chuyển đổi từ đơn vị micro giây

3.2. Phân tích xu hướng trên biểu đồ

Để trực quan hóa mối tương quan giữa khối lượng công việc và hiệu quả của kiến trúc đa nhân, chúng tôi biểu diễn dữ liệu trên biểu đồ (Hình 2). Trục hoành biểu thị số vòng lặp theo thang đo logarit cơ số 10, trục tung biểu thị hệ số tăng tốc theo thang đo tuyến tính.



Hình 2. Biểu đồ thể hiện hệ số tăng tốc theo số lượng vòng lặp. Đường biểu diễn tiệm cận mức giới hạn lý thuyết 2.0x khi tải trọng tăng lên

Qua biểu đồ, ta thấy rõ hệ số tăng tốc dần tiệm cận mức 2.0. Ở các mức tải thấp (dưới 10.000 vòng lặp), đường biểu diễn có độ dốc lớn, xuất phát từ mức 1.39x và tăng nhanh lên 1.85x. Điều này phản ánh giai đoạn mà chi phí khởi tạo tác vụ vẫn còn ảnh hưởng đáng kể đến tổng thời gian thực thi. Khi khối lượng công việc vượt qua ngưỡng 100.000 vòng lặp, đường biểu diễn đi ngang (bão hòa) và ổn định ở mức 1.99x. Tại điểm này, đường đồ thị gần như trùng khít với đường giới hạn lý thuyết (S=2), chứng tỏ hệ thống đã tiệm cận trạng thái cân bằng tải tối ưu.

3.3. Phân tích hiệu quả song song hóa và chi phí quản lý

Kết quả thực nghiệm từ nghiên cứu này đã làm sáng tỏ mối quan hệ phi tuyến tính giữa khối lượng công việc và hiệu suất tăng tốc trên nền tảng ESP32. Tại mức tải thấp nhất (1.000 vòng lặp), hệ số tăng tốc chỉ đạt 1.391x, thấp hơn đáng kể so với mức kỳ vọng lý thuyết là 2.0x. Hiện tượng này có thể được giải thích thông qua cơ chế quản lý tác vụ của hệ điều hành FreeRTOS. Để kích hoạt chế độ đa nhân, hệ thống phải thực hiện một loạt các thao tác nền tảng (overhead) bao gồm: cấp phát ngăn xếp (stack allocation), khởi tạo khối điều khiển tác vụ (TCB) và thực hiện chuyển đổi ngữ cảnh (context switching) (Barry, 2024). Khi thời gian tính toán thực tế quá ngắn (dưới 3 mili giây), các chi phí quản lý cố định này chiếm một tỷ trọng lớn trong tổng thời gian thực thi, dẫn đến việc làm giảm chỉ số hiệu suất song song xuống còn 69.5%.

Ngược lại, khi khối lượng tính toán tăng lên đến 10 triệu vòng lặp, thời gian thực thi kéo dài hơn 20 giây. Lúc này, chi phí khởi tạo ban đầu trở nên không đáng kể so với tổng thời gian xử lý. Hệ số tăng tốc đạt 1.995x (hiệu suất 99.8%) cho thấy kiến trúc Xtensa® LX6 Dual-Core có khả năng mở rộng tuyến tính gần như hoàn hảo đối với các tác vụ độc lập. Điều này

cũng xác nhận tính đúng đắn của chiến lược thiết kế thuật toán register-based compute-bound mà chúng tôi đã đề xuất ở Mục 3.2. Bằng cách hạn chế tối đa việc truy xuất bộ nhớ Flash ngoài và giữ dữ liệu nằm trong các thanh ghi, hai nhân CPU có thể hoạt động song song hết công suất mà không gặp phải hiện tượng nghẽn cổ chai tại bus dữ liệu hay tranh chấp bộ nhớ đệm.

Bên cạnh đó, nghiên cứu cũng làm nổi bật vai trò quan trọng của việc quản lý Watchdog Timer trong các ứng dụng hiệu năng cao. Việc tinh chỉnh thời gian chờ lên 60 giây đã chứng minh là giải pháp cần thiết để duy trì sự ổn định hệ thống, ngăn chặn các sự cố khởi động lại ngoài ý muốn khi CPU bị chiếm dụng hoàn toàn bởi các tác vụ điện toán biên (Edge Computing).

IV. Kết luận

Nghiên cứu này đã đánh giá toàn diện khả năng xử lý đa nhân của vi điều khiển ESP32-WROOM-32E-N4 thông qua các bài kiểm chuẩn tính toán dấu phẩy động. Kết quả cho thấy việc sử dụng kỹ thuật lập trình đa luồng (multi-threading) có thể giúp giảm một nửa thời gian xử lý cho các tác vụ nặng, biến ESP32 trở thành một ứng viên sáng giá cho các ứng dụng IoT đòi hỏi năng lực xử lý tại chỗ như xử lý tín hiệu số hay chạy các mô hình học máy đơn giản.

Từ những dữ liệu thu được, chúng tôi đưa ra các kiến nghị kỹ thuật sau cho việc phát triển ứng dụng trên nền tảng này: Thứ nhất, chỉ nên áp dụng mô hình đa nhân cho các tác vụ có thời gian thực thi ước tính lớn hơn 10 mili giây. Đối với các tác vụ quá nhỏ, chi phí quản lý của hệ điều hành sẽ triệt tiêu lợi ích của việc song song hóa, khiến việc chạy tuần tự trên đơn nhân trở thành lựa chọn hiệu quả hơn. Thứ hai, khi thiết kế các thuật toán song song, lập trình viên cần tối ưu hóa việc sử dụng thanh ghi và bộ nhớ RAM nội bộ, hạn chế tối đa việc truy xuất đồng thời vào bộ nhớ Flash để tránh xung đột đường truyền. Cuối cùng, việc cấu hình lại Task Watchdog Timer là bắt buộc khi triển khai các thuật toán tận dụng 100% tài nguyên CPU để đảm bảo hệ thống vận hành liên tục và tin cậy.

Tài liệu tham khảo

Amdahl, G. M. (1967). Validity of the single processor approach to achieving large scale computing capabilities. *Proceedings of the April 18-20, 1967, Spring Joint Computer Conference, AFIPS '67 (Spring)*, 483-485. https://doi.org/10.1145/1465482.1465560

Barry, R. (2024). *Mastering the FreeRTOS™ Real Time Kernel*. FreeRTOS (Amazon Web Services). https://www.freertos.org/Documentation/RTOS_book.html

ESP32-WROOM-32E & ESP32-WROOM-32UE Datasheet. (n.d.). Retrieved January 27, 2026, from https://documentation.espressif.com/api/resource/doc/file/Nyql9LzX/FILE/esp32-wroom-32e_esp32-wroom-32ue_datasheet_en.pdf

Shi, W., Cao, J., Zhang, Q., Li, Y., & Xu, L. (2016). Edge Computing: Vision and Challenges. *IEEE Internet of Things Journal*, *3*(5), 637-646. https://doi.org/10.1109/JIOT.2016.2579198

Watchdogs-ESP32--ESP-IDF Programming Guide v5.5.2 documentation. (n.d.). Retrieved January 27, 2026, from https://docs.espressif.com/projects/esp-idf/en/stable/esp32/api-reference/system/wdts.html#watchdogs

Williams, S., Waterman, A., & Patterson, D. (2009). Roofline: An insightful visual performance model for multicore architectures. *Communications of the ACM*, *52*(4), 65-76. https://doi.org/10.1145/1498765.1498785

ERFORMANCE ANALYSIS OF SYMMETRICAL MULTIPROCESSING (SMP) ON THE ESP32-WROOM-32E-N4 MICROCONTROLLER

Quach Thi Hanh[1], Nguyen Thi To Uyen[1]

Abstract: *This research focuses on evaluating the practical effectiveness of Symmetrical Multiprocessing (SMP) architecture on the ESP32-WROOM-32E-N4 microcontroller. Using an experimental methodology, the study compares the execution times of floating-point arithmetic algorithms across two scenarios involving sequential execution on a single core and parallel execution on dual cores. Experimental results demonstrate that for compute-bound tasks with a sufficiently large workload, leveraging the second core allows the system to achieve a speedup factor of 1.995, which is equivalent to 99.7% of the theoretical performance limit. However, the study also indicates that for tasks with short execution times of less than 4 milliseconds, the overhead associated with task creation and context management within the FreeRTOS real-time operating system significantly reduces parallelization efficiency. These findings provide a critical foundation for optimizing Edge Computing applications on low-cost IoT devices.*

Keywords: *dual-core microcontroller, ESP32, floating-point, performance benchmarking, FreeRTOS, symmetrical multiprocessing (SMP), Xtensa LX6*

[1] Faculty of Electric and Electronic Engineering, Hanoi Open University, Hanoi, Vietnam