CẬP NHẬT CHƯƠNG TRÌNH TỪ XA CHO THIẾT BỊ NHÚNG CÓ KẾT NỐI INTERNET

FIRMWARE OVER THE AIR FOR EMBEDDED DEVICE WITH INTERNET CONNECTION

Nguyễn Huỳnh Nhật Thương[1]*, Trần Thụy Ngọc Hằng[1], Nguyễn Hoàng Phương Trinh[1], Võ Tuấn Minh[2]

[1]Công ty TNHH Kỹ thuật TAPIT

[2]Trường Đại học Bách khoa – Đại học Đà Nẵng

*Tác giả liên hệ: nhatthuongqn@gmail.com

(Nhận bài: 11/12/2020; Chấp nhận đăng: 28/7/2021)

Tóm tắt - Cập nhật chương trình từ xa đang trở thành một chức năng không thể thiếu cho thiết bị nhúng có kết nối Internet. Bài báo này trình bày phương án triển khai, thiết kế phần mềm và phần cứng để thực hiện chức năng cập nhật chương trình từ xa. Kết quả nghiên cứu đã được thực nghiệm 600 lần, dựa trên hai định dạng tập tin chương trình phổ biến là HEX và BIN trên một thiết bị sử dụng vi xử lý ARM Cortex-M, kết nối Internet thông qua mạng 4G. Các đánh giá cho thấy, phương án thực hiện có độ tin cậy cao, tỉ lệ thực hiện thành công là 100%, toàn bộ tính năng thêm vào chỉ chiếm 20,28 KByte bộ nhớ của thiết bị, thời gian thực hiện cập nhật chỉ tốn 11,5 giây đối với một chương trình ứng dụng có kích cỡ 60KByte. Ngoài ra, hoạt động của thiết bị vẫn được đảm bảo khi các sự cố khách quan xảy ra trong quá trình cập nhật. Kết quả nghiên cứu cũng đã được áp dụng vào một sản phẩm hoàn thiện trên thị trường.

Từ khóa - Cập nhật chương trình từ xa; thiết bị nhúng; vi xử lý ARM Cortex-M; vi điều khiển STM32; vạn vật kết nối Internet

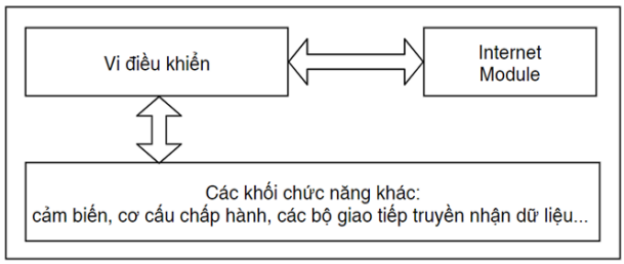
Abstract - Firmware Over The Air (FOTA) is an indispensable feature in most of Internet connected embedded devices. The major parts in this proposal include the deployment plan; hardware and software design, being built to implement the FOTA feature for an embedded device. This feature has been tested 600 times of transmitting files in two popular formats of BIN and HEX based on an experimental hardware device based on ARM Cortex-M microprocessor with Internet connection via the 4G network. Generally, the function of this feature has been completed and achieved certain measurement results with 100% success rate. The additional software process occupies 20.28 KByte. It takes 11.5 second per one updating time for a 60 KByte application program. Moreover, operation of the embedded device is still guaranteed in some failure cases during the updating process. The designed FOTA feature has been applied for the upgrading of an embedded device on the market.

Key words - Firmware over the air; embedded device; Internet of things; ARM Cortex-M Processor; STM32 Microcontroller

1. Giới thiệu

Cùng với sự phát triển không ngừng của công nghệ vi xử lý, vi điều khiển và đặc biệt là xu thế vạn vật kết nối Internet, ngày càng nhiều thiết bị nhúng kết nối Internet được sử dụng khắp nơi trên thế giới trong rất nhiều lĩnh vực khác nhau như công nghiệp, tự động hóa, điều khiển, quan trắc, truyền tin,… Điều này mở ra hàng loạt các cơ hội và thách thức mới cho các nhà nghiên cứu và phát triển. Thiết bị nhúng rất đa dạng, phong phú về chủng loại và mức độ phức tạp tùy vào công năng sử dụng. Thông thường, trong một thiết bị nhúng, vi điều khiển là nơi quản lý thực thi các chương trình. Có thể coi đấy là một máy tính nhỏ bị giới hạn về bộ nhớ và tốc độ xử lý. Ngoài vi điều khiển, để thiết bị nhúng có thể kết nối được với Internet thì cần có một Internet Module xử lý tín hiệu với chức năng thực hiện các kết nối 3G/4G, Wifi, hoặc Ethernet. Vi điều khiển sẽ kết nối và giao tiếp với module đó thông qua các ngoại vi giao tiếp được tích hợp sẵn. Nếu quan tâm thiết bị nhúng dưới góc độ tính năng cập nhật chương trình từ xa thì thiết bị sẽ được mô tả với ba thành phần cơ bản là vi điều khiển, Internet Module và các khối chức năng khác phục vụ cho những mục đích sử dụng cụ thể của thiết bị như Hình 1.

Trong bối cảnh chạy đua rất gắt gao giữa các nhà sản xuất, thời gian thiết kế và cung cấp thiết bị ra thị trường là rất ngắn. Do vậy, thường có xu hướng là các thiết bị được trang bị phần cứng có thiết kế dự phòng để đáp ứng được yêu cầu thị trường trong một khoảng thời gian dài còn phần mềm có thể được triển khai theo từng giai đoạn. Bên cạnh đó, nhiều thiết bị sau khi đưa vào sử dụng thì mới phát hiện các lỗi hay cần nâng cấp thêm để bổ sung, tối ưu các tính năng theo nhu cầu phát sinh. Tuy nhiên, có một thực tế là không ít thiết bị được lắp đặt sử dụng tại những nơi mà đội ngũ kỹ thuật khó có thể tiếp cận được, điều này dẫn đến những khó khăn trong quá trình thay đổi chương trình chạy trên thiết bị. Một nghiên cứu gần đây cho thấy, tần suất cập nhật chương trình sẽ tăng lên đáng kể trong những năm tới, thậm chí có khả năng cập nhật hàng tháng [1].



Hình 1. *Mô hình thiết bị nhúng có kết nối Internet*

Để giải quyết được những vấn đề nêu trên thì giải pháp cập nhật chương trình từ xa đang được quan tâm và phát triển như một tính năng quan trọng cho các thiết bị có kết nối Internet. Tính năng này giúp nhà sản xuất có thể cập nhật chương trình từ xa cho thiết bị nhúng bằng cách thay

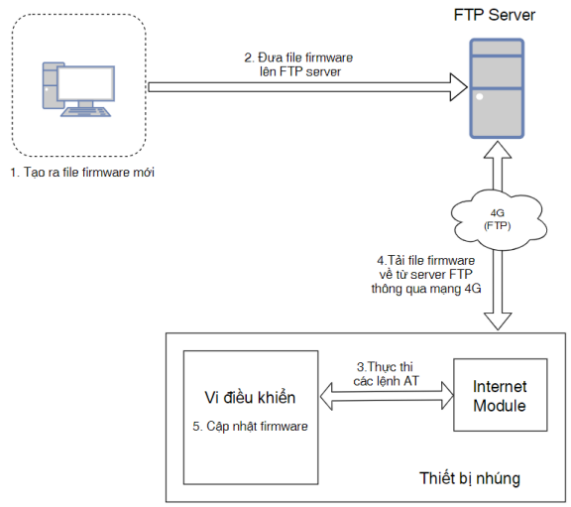
[1] TAPIT Engineering Co., Ltd (Thuong Nguyen, Tran Thuy Ngoc Hang, Nguyen Hoang Phuong Trinh)

[2] The University of Danang - University of Science and Technology (Vo Tuan Minh)

thế chương trình hiện tại trên vi điều khiển bằng một chương trình mới có chứa các thay đổi, bổ sung tính năng hoặc cập nhật những thiếu sót về bảo mật của thiết bị thông qua môi trường Internet một cách nhanh chóng và không cần thao tác trực tiếp lên thiết bị. Nghiên cứu này trình bày phương án triển khai, thiết kế phần mềm, phần cứng để thực hiện chức năng cập nhật chương trình từ xa và đưa ra các đánh giá khi thử nghiệm tính năng này trên thiết bị phần cứng cụ thể.

Trên thực tế, một số nền tảng Internet vạn vật cung cấp dịch vụ liên quan đến tính năng cập nhật chương trình từ xa để hỗ trợ người dùng, tuy nhiên các nền tảng này chỉ hỗ trợ cho một số phần cứng nhất định [2], [3]. Đối với những nhà sản xuất thiết bị vi điều khiển, họ cung cấp các tài liệu hướng dẫn đơn lẻ từng thành phần như xây dựng bootloader, triển khai in-application programming [4] cùng với các ví dụ thử nghiệm ở mức độ đơn giản và phụ thuộc vào các công cụ phần mềm đi kèm của hãng [5], [6]. Vì vậy, người dùng thiếu những phương án triển khai cập nhật chương trình từ xa cụ thể cùng với các so sánh đánh giá để phân tích, lựa chọn áp dụng cho dự án, phụ thuộc vào đơn vị cung cấp nền tảng, gặp khó khăn trong việc triển khai và quản lý cập nhật hàng loạt. Các phương án triển khai cập nhật chương trình cụ thể được nêu ra và thử nghiệm trong bài báo này cùng với các phân tích giúp người dùng có thể tiết kiệm thời gian, công sức, có thêm sự tham chiếu để lựa chọn được giải pháp phù hợp trong quá trình phát triển sản phẩm, làm chủ chương trình ứng dụng và không phụ thuộc vào nền tảng Internet vạn vật.

2. Phương án triển khai



Hình 2. *Quá trình cập nhật chương trình từ xa*

Trong nghiên cứu này, việc triển khai cập nhật chương trình từ xa cho thiết bị nhúng được đề xuất tiến hành theo từng bước như sau: (1) Khi nhà sản xuất muốn thực hiện cập nhật chương trình từ xa, người phát triển chương trình sẽ tạo ra chương trình ứng dụng mới cho vi điều khiển trong thiết bị nhúng; (2) Tập tin chứa chương trình ứng dụng mới được tải lên và lưu trữ trên một máy chủ quản lý tập tin chương trình (File Transfer Protocol Server); (3) Mặt khác, một cách định kì, trong thiết bị nhúng, vi điều khiển gửi các lệnh để Internet Module thực thi việc kiểm tra phiên bản mới của chương trình ứng dụng trên máy chủ; (4) Nếu tồn tại tập tin chương trình phiên bản mới trên máy chủ FTP, Internet Module sẽ thực thi quá trình tải tập tin này về vi điều khiển theo giao thức FTP. Ở bước kế tiếp, vi điều khiển sẽ thay thế chương trình ứng dụng đang chạy bằng chương trình ứng dụng mới được đưa ra từ nhà sản xuất. Quá trình cập nhật chương trình từ xa được thể hiện ở Hình 2.

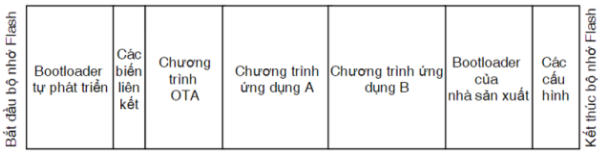
3. Đề xuất thiết kế

3.1. Thiết kế phần mềm

Chức năng cập nhật chương trình từ xa phải được xây dựng trên nguyên tắc đảm bảo các tiêu chí: (i) Tỉ lệ thực hiện thành công tuyệt đối trong môi trường thử nghiệm với các tình huống giả định xảy ra trong thực tế; (ii) Sử dụng tài nguyên bộ nhớ ít; (iii) Quá trình cập nhật chương trình từ xa diễn ra trong thời gian ngắn, thường là trong thời gian mà hệ thống ở trạng thái chờ (idle); (iv) Hoạt động của thiết bị được đảm bảo trong các trường hợp như mất nguồn, mất kết nối Internet, xuất hiện lỗi trong quá trình cập nhật chương trình từ xa, hoặc phiên bản chương trình mới được cập nhật tồn tại lỗi. Để đạt được các tiêu chí trên thì phần mềm được thiết kế như sau:

3.1.1. Phân vùng bộ nhớ chương trình

Thông thường, một chương trình ứng dụng sẽ được lưu từ địa chỉ bắt đầu của bộ nhớ Flash. Khi khởi động, vi điều khiển sẽ thực thi chương trình tại vị trí này. Tuy nhiên, để phát triển chức năng cập nhật chương trình từ xa thì bộ nhớ Flash sẽ được chia thành nhiều phân vùng khác nhau để lưu trữ các chương trình. Trong nghiên cứu này, bộ nhớ Flash được chia thành các phân vùng sau để lưu trữ: Bootloader tự phát triển, vùng nhớ chứa các biến liên kết, chương trình OTA và hai phân vùng chương trình ứng dụng. Trong đó, chương trình Bootloader tự phát triển sẽ nằm ở vị trí đầu tiên của bộ nhớ chương trình. Mỗi khi vi điều khiển khởi động, thông qua giá trị các biến liên kết, Bootloader tự phát triển sẽ quyết định lựa chọn thực thi chương trình OTA hay chương trình ứng dụng nào trong lần khởi động này. Vùng nhớ các biến liên kết chứa thông tin chương trình cần được thực thi và thông tin phiên bản chương trình hiện tại. Chương trình OTA có nhiệm vụ chính là thực hiện tải tập tin chương trình ứng dụng từ máy chủ quản lý tập tin và thực hiện thay thế chương trình ứng dụng. Chương trình ứng dụng là chương trình thực hiện các chức năng chính của thiết bị nhúng. Hình 3 thể hiện các phân vùng trong bộ nhớ Flash của vi điều khiển.



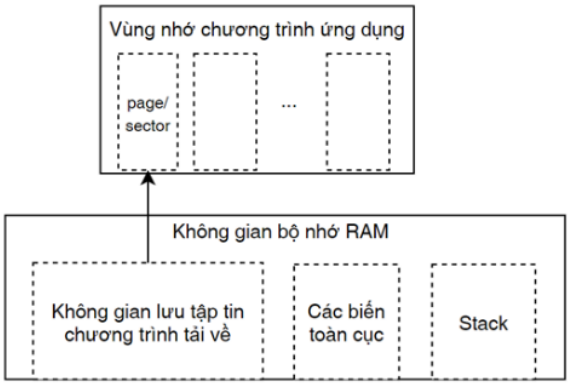
Hình 3. *Các phân vùng trong bộ nhớ Flash*

Với cách phân chia sử dụng bộ nhớ này thì có hai phân vùng dành cho chương trình ứng dụng, nghĩa là trong vi điều khiển tồn tại hai phiên bản chương trình ứng dụng khác nhau. Người phát triển có thể thiết kế một phân vùng (A/B) dành cho phiên bản xuất xưởng của ứng dụng, một phân vùng còn lại (B/A) để chạy chương trình chính và các bản cập nhật. Ngoài ra, người phát triển có thể thiết kế luân

phiên cập nhật chương trình cho các vị trí A/B để thiết bị luôn duy trì hai phiên bản chương trình mới nhất. Thiết kế này cũng đảm bảo luôn có một chương trình ứng dụng dự phòng trong các tình huống: Mất Internet trong quá trình cập nhật chương trình hoặc bản cập nhật mới tồn tại lỗi làm chương trình không hoạt động được [7], [8].

3.1.2. Phương pháp lưu dữ liệu tạm thời

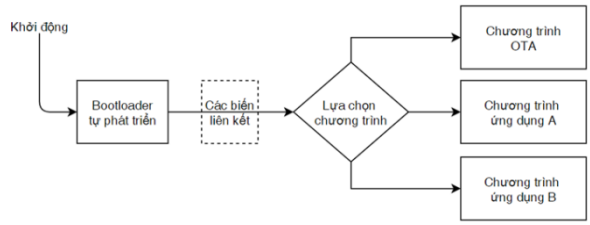
Trong quá trình cập nhật FOTA, mỗi khi thiết bị nhúng nhận được một gói tin chứa một phần hoặc toàn bộ nội dung của tập tin chương trình ứng dụng từ máy chủ quản lý tập tin thì gói tin này sẽ được lưu ở bộ nhớ dữ liệu (RAM) trước khi được ghi vào bộ nhớ Flash như mô tả ở Hình 4. Tùy thuộc vào tài nguyên phần cứng, thuật toán cho phép cấu hình để lựa chọn kích cỡ dữ liệu mỗi lần tải về và kích cỡ bộ đệm dữ liệu tại RAM. Mỗi khi nhận một gói dữ liệu chứa chương trình ứng dụng, ghi phần dữ liệu này vào vùng nhớ chứa chương trình ứng dụng tại bộ nhớ Flash. Một tập tin chương trình có thể sẽ được tải một lần hoặc nhiều lần tùy thuộc vào kích cỡ của tập tin so với kích cỡ bộ nhớ đệm. Nếu bộ đệm được khai báo nhỏ hơn so với kích cỡ của tập tin thì phương pháp lưu trữ dữ liệu tạm thời là lưu trữ một phần. Nếu bộ đệm được khai báo lớn hơn hoặc bằng so với kích cỡ tập tin chương trình ứng dụng thì phương pháp lưu trữ dữ liệu tạm thời là lưu trữ toàn phần [9].



Hình 4. *Tổ chức bộ nhớ dữ liệu*

3.1.3. Thay đổi chương trình trong thiết bị

Từ thiết kế phân vùng bộ nhớ chương trình và phương pháp lưu trữ dữ liệu tạm thời, quá trình thay đổi chương trình trong thiết bị được thiết kế như sau: Chương trình ứng dụng kết nối và kiểm tra thông tin phiên bản chương trình trên máy chủ một cách định kì. Nếu không có phiên bản mới, chương trình ứng dụng tiếp tục thực hiện bình thường. Khi có phiên bản mới trên máy chủ, chương trình ứng dụng sẽ thay đổi giá trị các biến liên kết và gọi lệnh khởi động lại vi điều khiển. Lúc này, Bootloader tự phát triển sẽ được thực thi và kiểm tra các biến liên kết, sau đó nhảy tới chương trình OTA. Chương trình OTA sẽ thực hiện cập nhật chương trình ứng dụng, sau khi quá trình này hoàn thành, giá trị biến liên kết được thay đổi và vi điều khiển sẽ được khởi động lại. Chương trình Bootloader tự phát triển tiếp tục được thực thi và kiểm tra biến liên kết để nhảy đến chương trình ứng dụng mới. Sơ đồ Hình 5 biểu diễn quá trình lựa chọn chương trình để thực thi trong thiết bị. Tuy nhiên, trong trường hợp xảy ra lỗi hoặc mất kết nối trong quá trình cập nhập, thiết bị sẽ được khởi động và chạy lại chương trình ứng dụng phiên bản cũ. Đồng thời, giá trị của các biến liên kết sẽ không thay đổi.



Hình 5. *Quá trình lựa chọn chương trình thực thi*

3.1.4. Webserver quản lý cập nhật chương trình từ xa

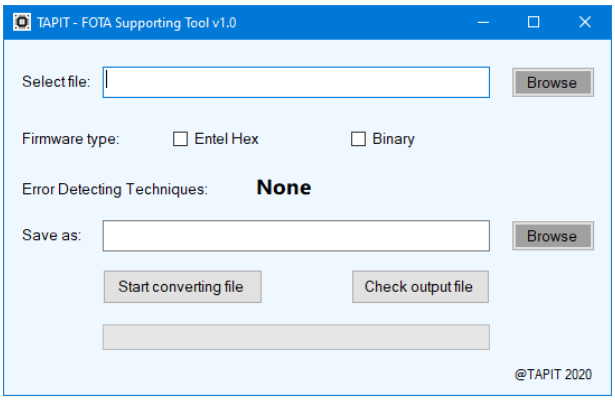
Tập tin chứa chương trình ứng dụng mới phục vụ cho quá trình cập nhật chương trình từ xa được đưa lên máy chủ quản lý tập tin (FTP server) tại địa chỉ http://fota.tapit.com. Webserver có giao diện như Hình 6 được thiết kế giúp quản lý quá trình cập nhật một cách hiệu quả với các tính năng: Cấu hình thiết bị hoặc nhóm thiết bị và phiên bản cập nhật mong muốn thông qua mã thiết bị (ID) và tên phiên bản chương trình. Ngoài ra, trạng thái cập nhập chương trình của từng thiết bị được giám sát thông qua một bảng gồm các thông số ID thiết bị, trạng thái thiết bị, phiên bản chương trình ứng dụng hiện tại của thiết bị và thời gian thiết bị cập nhập trạng thái lên hệ thống quản lý.

Hình 6. *Giao diện webserver quản lý thiết bị*

3.1.5. Kiểm tra toàn vẹn dữ liệu

Việc kiểm tra toàn vẹn dữ liệu tại chương trình OTA trước khi thay đổi giá trị biến liên kết và khởi động lại vi điều khiển cần được thực hiện để tránh trường hợp xuất hiện lỗi trong nội dung chương trình mới. Thiết kế chương trình OTA cần có tính năng kiểm tra toàn vẹn dữ liệu cho cả hai loại định dạng tập tin phổ biến HEX và BIN. Trong đó, tập tin định dạng HEX đã được thêm mã checksum để kiểm tra toàn vẹn dữ liệu [10]. Sau khi một phần tập tin hoặc toàn bộ tập tin được tải về bộ đệm dữ liệu tại RAM, thuật toán checksum được xây dựng để kiểm tra từng hàng dữ liệu trước khi ghi vào bộ nhớ Flash. Đối với tập tin định dạng BIN, nội dung tập tin chỉ chứa chương trình ứng dụng mà không có sẵn một giải pháp để kiểm tra toàn vẹn dữ liệu. Vì vậy người phát triển cần thiết kế và tích hợp giải pháp kiểm tra toàn vẹn để lựa chọn tập tin phù hợp để hỗ trợ cho quá trình cập nhập. Hiện nay, hầu hết các dòng vi điều khiển

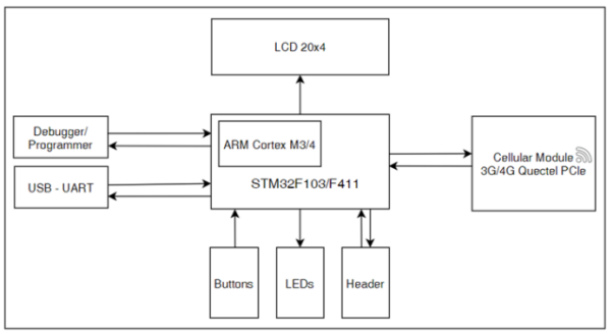
STM32 đều tích hợp sẵn khối phần cứng tính toán Cyclic Redundancy Check (CRC). Kiểm tra toàn vẹn dữ liệu sử dụng CRC được lựa chọn thiết kế cho tập tin định dạng BIN để hỗ trợ cho việc kiểm tra toàn vẹn dữ liệu trong nghiên cứu cập nhập chương trình từ xa cho thiết bị nhúng [11]. Giá trị CRC cần được tính toán và thêm vào ở cuối mỗi tập tin trước khi đưa lên máy chủ. Phần mềm TAPIT_FOTA_Supporting_Tool_v1.0.exe có giao diện như Hình 7 được thiết kế để hỗ trợ cho việc tính toán giá trị CRC và bổ sung vào tập tin. Sau khi toàn bộ tập tin được tải về và ghi vào bộ nhớ Flash, chương trình sẽ sử dụng khối CRC để kiểm tra toàn vẹn dữ liệu cho chương trình mới.



Hình 7. *Giao diện phầm mềm FOTA Supporting Tool*

3.2. Thiết kế phần cứng thử nghiệm

Trong nghiên cứu này, phần cứng được thiết kế để phục vụ cho quá trình thử nghiệm tính năng cập nhật chương trình từ xa. Theo thống kê năm 2019 của SoftBank, trong lĩnh vực hệ thống nhúng, số lượng thiết bị IoTs sử dụng lõi ARM chiếm 90% thị phần và số lượng vi điều khiển trên thế giới được thiết kế trên kiến trúc lõi ARM chiếm 25% thị phần [12], trong đó dòng Cortex-M chiếm tỉ lệ cao nhất. Vì vậy, thiết bị phần cứng được xây dựng sử dụng một vi điều khiển 32bit của hãng STMicroelectronics sử dụng CPU lõi ARM Cortex-M, với cấu hình 512 Kbytes bộ nhớ Flash và 128 Kbytes bộ nhớ RAM, được tích hợp sẵn ngoại vi thời gian thực RTC, và khối ngoại vi tính toán mã CRC để kiểm tra tính toàn vẹn dữ liệu.



Hình 8. *Sơ đồ khối thiết bị thử nghiệm*

Module EC21 của hãng Quectel được lựa chọn làm Internet Module để thiết bị có thể kết nối vào Internet [13], Module này hỗ trợ công nghệ 3G/4G. Một nút nhấn được thiết kế kết nối với vi điều khiển để có thể gửi yêu cầu thực hiện cập nhật chương trình từ xa ngay lập tức mỗi khi nút được nhấn. Các thông báo, trạng thái làm việc của thiết bị được thể hiện thông qua một màn hình LCD và các đèn LED trên bảng mạch. Bên cạnh đó, bảng mạch cũng được thiết kế hỗ trợ kết nối với máy tính để nạp các chương trình thử nghiệm và thực hiện gỡ lỗi thông qua cổng Debugger/Programmer. Cổng USB-UART được thiết kế để thiết bị có thể gửi nhật ký hoạt động, trạng thái lên máy tính để lưu trữ, phục vụ cho việc đánh giá khi thực hiện thử nghiệm quá trình cập nhật chương trình nhiều lần. Sơ đồ khối của phần cứng được trình bày ở Hình 8 và hình ảnh thực tế của thiết bị thử nghiệm được trình bày ở Hình 9.

Hình 9. *Thiết bị phần cứng thử nghiệm*

4. Đánh giá kết quả thực nghiệm

Sau khi hoàn thành các thiết kế phần mềm và phần cứng, nhóm nghiên cứu đã tiến hành thực nghiệm để đánh giá kết quả nghiên cứu dựa trên nguyên tắc đảm bảo ba tiêu chí: (i) Định lượng bằng tài nguyên bộ nhớ khi triển khai tính năng FOTA; (ii) Tỉ lệ thành công khi thực hiện FOTA; (ii) Thời gian thực hiện quá trình FOTA. Kết quả thực nghiệm được đánh giá chi tiết theo 3 tiêu chí trên như sau:

4.1. Chiếm dụng bộ nhớ

Từ kết quả thông báo của trình biên dịch, kết hợp với tập tin chứa thông tin về quá trình liên kết và định vị các thành phần trong bộ nhớ của vi điều khiển (Linker Map File) ta có các thông tin về việc chiếm dụng bộ nhớ của các chương trình (Bootloader và OTA) đối với bộ nhớ Flash và bộ nhớ SRAM như sau: Chương trình Bootloader và chương trình OTA chiếm dụng lượng bộ nhớ tổng cộng 32.53KByte trong bộ nhớ Flash của vi điều khiển cho trường hợp cập nhật chương trình từ xa sử dụng tập tin định dạng HEX hoặc 32.64KByte cho trường hợp cập nhật chương trình từ xa sử dụng tập tin định dạng BIN. Tuy nhiên, nếu trình biên dịch được cấu hình tối ưu về kích cỡ chương trình (code size) thì chương trình Bootloader tự xây dựng chiếm 5.88KByte và chương trình OTA chiếm 14.5KByte cho định dạng HEX, 14.4KByte cho định dạng BIN. Đây là một phần nhỏ so với tổng dung lượng bộ nhớ Flash của hầu hết các vi điều khiển lõi ARM Cortex-M hiện nay. Đối với bộ nhớ RAM, chương trình Bootloader chiếm 1.71KByte và chương trình OTA chiếm 2.4KByte. Trong đó, kích cỡ BufferDownload do người dùng tự khai báo tùy thuộc vào phần cứng của từng loại vi điều khiển.

4.2. Tỉ lệ thành công

Sau khi hoàn thành thiết kế phần cứng và chương trình thực nghiệm sử dụng thư viện cập nhật chương trình từ xa thì tỉ lệ cập nhật chương trình đã được kiểm thử với 03 chương trình ứng dụng với 03 kích cỡ khác nhau là 15KByte, 30KByte và 60KByte. Mỗi chương trình được biên dịch và tạo thành 1 tập tin dạng HEX và 1 tập tin dạng BIN. Kích cỡ của các tập tin chương trình được trình bày ở Bảng 1.

Bảng 1. *Kích cỡ các tập tin chương trình ứng dụng*

Kích cỡ chương trình ứng dụng	Kích cỡ tập tin chương trình định dạng BIN	Kích cỡ tập tin chương trình định dạng HEX
15KByte	16KByte	43KByte
30KByte	31KByte	85KByte
60KByte	61KByte	169KByte

Mỗi tập tin được thử nghiệm 100 lần và cho kết quả tỉ lệ cập nhật thành công là 100% như thể hiện ở Bảng 2. Kết quả này cho thấy, độ tin cậy tuyệt đối của thư viện xây dựng được trong việc cập nhật thành công trong môi trường thử nghiệm, có thể áp dụng được vào thực tế trên thiết bị Datalogger.

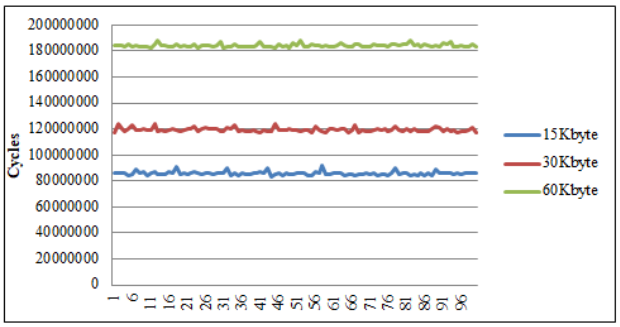
Bảng 2. *Tỉ lệ thực hiện cập nhật chương trình thành công*

Kích cỡ tập tin (KByte)	Số lần thử nghiệm	Số lần thành công	Tỉ lệ thành công
15	200	200	100%
30	200	200	100%
60	200	200	100%

4.3. Thời gian cập nhật

Vi điều khiển trên thiết bị thử nghiệm được cấu hình hoạt động với tốc độ vi xử lý 16MHz, ngoại vi giao tiếp Universal Asynchronous Receive/Transmit (UART) giao tiếp với Module Internet EC21 với tốc độ truyền nhận dữ liệu (baudrate) là 115200bps, khung truyền sử dụng là 1 bit bắt đầu, 8 bit dữ liệu, không sử dụng bit kiểm tra lỗi (parity bit) và 1 bit kết thúc.

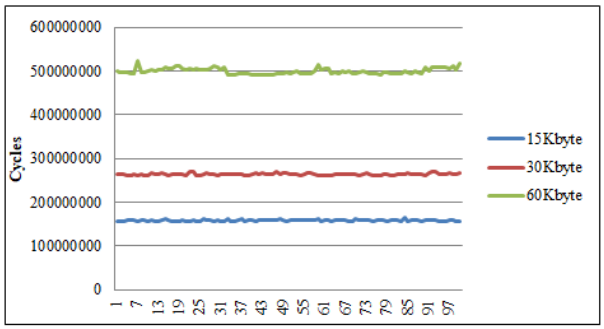
Khối Data Watch and Trace (DWT) bên trong vi xử lý ARM Cortex - M được sử dụng để giám sát số xung đồng hồ (clock cycles) khi vi xử lý thực hiện cả quá trình cập nhật chương trình từ xa và các quá trình thành phần, bao gồm: Thời gian kiểm tra cường độ tín hiệu sóng (RSSI) của Module EC21, thời gian kết nối Internet, thời gian kết nối đến máy chủ quản lý tập tin, thời gian tải – ghi tập tin vào bộ nhớ Flash, thời gian ngắt kết nối với máy chủ - ngắt Internet.



Hình 10. *Kết quả clock cycles đối với các tập tin dạng BIN*

Kết quả đo đạt được sau khi tiến hành thử nghiệm trên các tập tin định dạng BIN trong 100 lần thử nghiệm được thể hiện ở Hình 10. Có thể thấy được tập tin có kích cỡ càng lớn thì quá trình cập nhật từ xa càng tốn nhiều số xung đồng hồ.

Kết quả đo đạt được sau khi tiến hành thử nghiệm trên các tập tin định dạng HEX trong 100 lần thử nghiệm được thể hiện ở Hình 11. Tương tự với các tập tin định dạng BIN, tập tin định dạng HEX có kích cỡ càng lớn thì quá trình cập nhật từ xa càng tốn nhiều số xung đồng hồ. Và cùng một kích cỡ chương trình ứng dụng, tập tin định dạng HEX tốn nhiều số xung đồng hồ hơn để thực hiện quá trình cập nhật so với tập tin định dạng BIN.



Hình 11. *Kết quả clock cycles đối với các tập tin dạng HEX*

Với tốc độ vi xử lý là 16MHz, số xung đồng hồ để thực hiện quá trình cập nhật chương trình từ xa ứng với từng tập tin chương trình được tính toán thành thời gian ở đơn vị giây và trình bày tại Bảng 3.

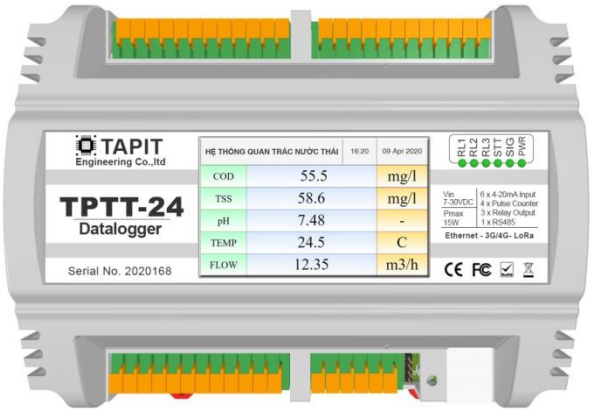
Bảng 3. *Tổng hợp thời gian cập nhật chương trình*

Kích cỡ chương trình ứng dụng	Định dạng BIN		Định dạng HEX	
	Cycles	Thời gian (s)	Cycles	Thời gian (s)
15Kbyte	85840094	5,4	158237974	9,9
30Kbyte	119395522	7,5	264065064	16,5
60Kbyte	184158711	11,5	499840662	31,2

Kết quả thống kê thời gian các quá trình thành phần cho thấy thời gian tải và ghi chương trình vào bộ nhớ Flash đối với các tập tin định dạng BIN với kích cỡ chương trình 15KByte, 30KByte, 60KByte chiếm tỉ lệ phần trăm lần lượt 67%, 76%, 76% trong tổng thời gian. Đối với tập tin chương trình định dạng HEX, thời gian thành phần này lần lượt chiếm 82%, 89% và 94% tổng thời gian. Vậy thời gian thực hiện cập nhật chương trình từ xa phụ thuộc nhiều vào thời gian tải tập tin và ghi chương trình vào bộ nhớ FLASH của vi điều khiển. Với cùng một chương trình, tập tin định dạng BIN chỉ chứa nội dung chương trình nên kích thước nhỏ làm cho thời gian tải nhanh hơn, tập tin định dạng HEX có thêm các trường chỉ dẫn nên kích thước lớn, dẫn đến thời gian tải lâu hơn.

Kết quả của nghiên cứu đã được áp dụng thực tế trên thiết bị datalogger TPTT-24 của Công ty TNHH Kỹ thuật TAPIT. Thiết bị này sử dụng vi điều khiển STM32F303 với 256KByte bộ nhớ Flash, 48KByte bộ nhớ SRAM và sử dụng Module kết nối Internet Quectel EC21 - công nghệ mạng 4G. Thiết bị Datalogger TPTT-24 này được nâng cấp dựa trên một thiết bị Datalogger thu nhập, điều khiển và

truyền dữ liệu tự động [14] được phát triển trước đây của công ty TNHH Kỹ thuật TAPIT, nay được tích hợp thêm chức năng cập nhật chương trình từ xa.



Hình 12. *Thiết bị áp dụng kết quả nghiên cứu*

5. Kết luận

Trong bài báo này, nhóm tác giả đã đề xuất và mô tả thiết kế tính năng cập nhật chương trình từ xa cho thiết bị nhúng có kết nối Internet trên một mô hình phần cứng cụ thể với vi điều khiển STM32 lõi ARM Cortex-M. Tuy nhiên, phương án triển khai được đề xuất trong bài báo có thể áp dụng cho các vi điều khiển có cấu trúc phần cứng khác nhau.

Kết quả thử nghiệm cho thấy, chương trình phần mềm thực hiện tính năng FOTA có độ tin cậy và tỉ lệ thành công khi thực hiện cập nhật chương trình từ xa là 100% trong 600 lần thử nghiệm, ít tốn kém tài nguyên bộ nhớ, phù hợp với nhiều thiết bị có cấu hình bộ nhớ khác nhau, quá trình cập nhật chương trình từ xa diễn ra trong thời gian ngắn, thời gian tải tập tin chương trình định dạng BIN nhanh hơn so với tập tin chương trình định dạng HEX với cùng một chương trình ứng dụng; thuật toán thiết kế đảm bảo thiết bị vẫn sử dụng được trong các trường hợp: Mất nguồn trong quá trình cập nhật chương trình từ xa, phiên bản chương trình mới tồn tại lỗi. Kết quả của nghiên cứu đã được áp dụng thực tế cho thiết bị Datalogger tại công ty TNHH Kỹ thuật TAPIT. Bên cạnh đó, trong quá trình nghiên cứu, nhóm tác giả nhận thấy được tầm quan trọng của việc bảo mật cho hệ thống FOTA. Vì thế, nhóm tác giả sẽ tiếp tục nghiên cứu về việc áp dụng giải thuật bảo mật cho quá trình cập nhật chương trình từ xa.

TÀI LIỆU THAM KHẢO

[1] Guissouma, Houssem, et al. "An empirical study on the current and future challenges of automotive software release and configuration management", *2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA). IEEE*, 2018, pp. 298-305, doi: 10.1109/SEAA.2018.00056.

[2] Thingsboard. "ESP32 OTA using ThingsBoard", https://thingsboard.io/

[3] Blynk. "Firmware Over-The-Air", https://docs.blynk.io/

[4] STMicroelectronics. "STM32 in-application programming (IAP) using the USART", https://www.st.com/

[5] Matej Sychra. "Http Update for ESP32" Arduino Reference, www.arduino.cc

[6] Texas Instruments. "SimpleLink™ Wi-Fi® CC3x20, CC3x3x Over-the-Air Update", www.ti.com

[7] Nick Lethaby. *"A more secure and reliable OTA update architecture for IoT devices"*, Texas Instruments, 2018.

[8] Koen Zandberg, Kaspar Schleiser, Francisco Acosta, Hannes Tschofenig, Emmanuel Baccelli. "Secure Firmware Updates for Constrained IoT Devices Using Open Standards: A Reality Check". *IEEE Access, IEEE*, 2019, 7, pp. 71907 - 71920. ff10.1109/ACCESS.2019.2919760ff. ffhal-02351794

[9] Brown, Benjamin Bucklin. *"Over-the-Air (OTA) Updates in Embedded Microcontroller Applications: Design Trade-Offs and Lessons Learned"*, Analog Devices, 2018.

[10] ARM, General: Intel hex file format, https://www.keil.com/

[11] STMicroelectronics, STM32F411xC STM32F411xE Datasheet - production data, https://www.st.com/

[12] ARM, Roadshow Slides Q1 2020, https://group.softbank/

[13] Quectel, Quectel EC21 LTE Standard Specification, https://www.quectel.com/

[14] Nguyễn Huỳnh Nhật Thương, Lê Công Vĩnh Khải, Ngô Đình Thanh, Phạm Văn Tuấn. "Thiết bị Datalogger thu nhập, điều khiển và truyền dữ liệu tự động", *Hội nghị Quốc gia lần thứ XXII về Điện tử, Truyền thông và Công nghệ thông tin (REV-ECIT)*, Hội Vô tuyến – Điện tử Việt Nam, 2019, pp.277-282.