

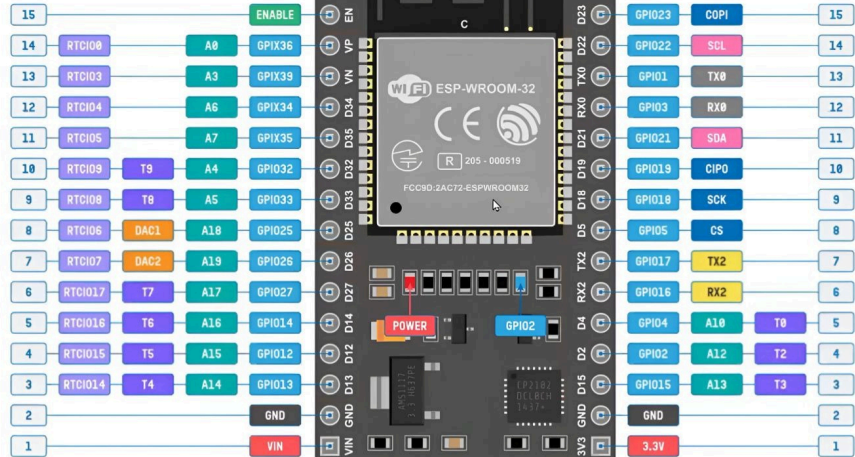
KIT ESP32 2



Board Esp32 dev kit v1

PHYSICAL PIN	POSITIVE SUPPLY
CONTROL PINS	GROUND SUPPLY
GPIO PINS	ADC INPUTS
DAC OUTPUTS	SPI PINS
TOUCH INPUTS	UART PINS
I2C PINS	EXCLUDED PINS

- GPIO pins 34, 35, 36 and 39 are input only.
- TX0 and RX0 (Serial0) are used for serial programming.
- TX2 and RX2 can be accessed as Serial2.
- Default SPI is VSPI. Both VSPI and HSPI pins can be set to any GPIO pins.
- All GPIO pins support PWM and interrupts.
- Built-in LED is connected to GPIO2.
- Some GPIO pins are used for interfacing flash memory and thus are not shown.



Copyright by Điện thông minh E-smart

Câu hỏi:

1. Code arduino nạp vào esp32 thông qua cổng micro usb đã được mã hoá thành mã nhị phân 0 1 và đóng gói để đưa vào chip u a r t?
2. Trong chip u a r t, có trình biên dịch mã nhị phân thành code của arduino?
3. Code arduino sau khi được biên dịch trở lại sẽ nạp vào chip flash để lưu giữ các tác vụ hành động?

Trả lời:

Dựa trên những kiến thức mà chúng ta vừa trao đổi, tôi xin được đính chính lại một vài điểm quan trọng để bạn nắm chắc hơn về quy trình này. Có một số hiểu lầm nhỏ về vai trò của chip U A R T và quá trình biên dịch:

1. Phân tích từng điểm của bạn:

Ý 1 (Sai): Code Arduino trên máy tính được mã hóa (biên dịch) thành mã nhị phân (0 và 1). Tuy nhiên, nó không “đóng gói” để đưa vào chip U A R T theo nghĩa dữ liệu thay đổi cấu trúc, mà chip U A R T chỉ đóng vai trò là “người vận chuyển” (cầu nối) để đưa dữ liệu đó từ cổng USB vào con E S P 32.

Ý 2 (Sai): Chip U A R T không có trình biên dịch. Nó là một con chip rất đơn giản, chỉ có nhiệm vụ chuyển đổi hình thức

tín hiệu: từ tín hiệu USB (máy tính hiểu) sang tín hiệu nhị thập U A R T (vi điều khiển hiểu). Quá trình biên dịch chuyển ra duy nhất 1 lần trên máy tính của bạn (thông qua phần mềm Arduino IDE).

Ý 3 (Sai về quy trình): Như đã nói ở trên, mã được nạp vào Flash luôn luôn là mã nhị phân (0 và 1). Chip Flash không bao giờ chứa “code Arduino” (ngôn ngữ C++ mà bạn viết). Nó chỉ chứa các lệnh mã máy để ESP 32 có thể thực thi trực tiếp.

2. Quy trình đúng 100% sẽ là:

Tại máy tính: Bạn viết code Arduino → Bấm nút “Upload” → Máy tính biên dịch code đó thành file Binary (nhị phân 0, 1).

Qua dây cáp USB: File nhị phân này được gửi đi dưới dạng tín hiệu USB.

Tại chip U A R T (trên bo mạch): Con chip này tiếp nhận tín hiệu USB, chuyển nó thành tín hiệu U A R T và đẩy sang các chân RX/TX của con ESP 32.

Tại con ESP 32: Chương trình bootloader (nằm trong ROM) sẽ “hứng” dòng mã nhị phân đang chảy vào từ chip U A R T và ghi trực tiếp dòng 0, 1 đó vào con chip Flash.

Khi hoạt động: ESP 32 đọc mã nhị phân từ Flash, nạp vào S RAM và chạy.

Tóm tắt để bạn dễ nhớ:

Máy tính: Là nơi “dịch thuật” (từ tiếng người sang tiếng máy).

Chip U A R T: Là “người đưa thư” (chuyển tín hiệu).

Chip Flash: Là “quyển vở” (lưu trữ mã nhị phân đã dịch).

ESP 32: Là “người thực thi” (đọc mã nhị phân và làm theo).

...

Câu hỏi:

Sao không truy cập trực tiếp vào chip flash luôn mà phải thông qua chip u a r t?

Trả lời:

Câu hỏi của bạn rất hay! Đây là thắc mắc chung của rất nhiều người khi mới tiếp cận điện tử. Lý do chính nằm ở việc “bắt đầu ngôn ngữ” và “bắt đầu phương thức kết nối” giữa máy tính và con chip nhớ Flash.

Dưới đây là 3 lý do chính tại sao chúng ta không thể truy cập trực tiếp:

1. Sự khác biệt về công kết nối (Hardware Interface)

Máy tính: Sử dụng chuẩn USB (Universal Serial Bus). Công USB

có 4 hoặc 5 dây, hoạt động với các giao thức phần mềm cực kỳ phức tạp.

Chip Flash: Thường sử dụng chuẩn SPI (Serial Peripheral Interface). Đây là một cách giao tiếp khác hoàn toàn, đòi hỏi các chân tín hiệu riêng biệt như MOSI, MISO, SCK.

Vấn đề: Bạn không thể cắm trực tiếp dây USB vào các chân của chip Flash vì chúng không cùng “ngôn ngữ vật lý”.

2. Vai trò của “Người trung gian” U A R T

Con chip U A R T (như CP2102 hay CH340 trên mạch) đóng vai trò là một phiên dịch viên.

Nó “nghe” tín hiệu từ cổng USB của máy tính.

Nó chuyển đổi dữ liệu đó sang chuẩn nối tiếp đơn giản là U A R T (chỉ cần 2 dây RX và TX).

Mọi vi điều khiển như E S P 32 hay Arduino đều có “tai” để nghe chuẩn U A R T này một cách dễ dàng.

3. Ai là người thực hiện việc “ghi” vào Flash?

Đây là điểm quan trọng nhất: Chip Flash là một bộ nhớ thụ động. Nó không tự biết cách nhận dữ liệu từ USB rồi tự lưu vào chính nó.

Để ghi dữ liệu vào Flash, cần một “người thợ” có trí thông minh để điều khiển. “Người thợ” đó chính là con chip E S P 32 (thông qua chương trình bootloader trong ROM mà chúng ta đã nói).

Máy tính gửi dữ liệu qua chip U A R T → E S P 32 nhận dữ liệu này → E S P 32 dùng các chân chuyên dụng để “viết” dòng dữ liệu đó vào chip Flash.

Nếu muốn nạp trực tiếp thì có được không?

Câu trả lời là Có, nhưng bạn sẽ cần một thiết bị chuyên dụng gọi là “Máy nạp ROM” (External Programmer).

Bạn phải tháo chip Flash ra khỏi mạch (hoặc dùng kẹp chuyên dụng).

Cắm chip Flash vào máy nạp đó. Máy nạp này sẽ đóng vai trò thay thế cho chip U A R T và E S P 32 để ghi thông dữ liệu vào.

Cách này cực kỳ mất thời gian và chỉ dùng khi chip bị lỗi nặng không thể nạp qua phần mềm được.

Tóm lại: Việc dùng chip U A R T là cách tiện lợi nhất, rồi mới đến máy tính có thể “nói chuyện” với vi điều khiển và nhờ vi điều khiển ghi dữ liệu vào Flash mà không cần tháo lắp linh kiện.

...

Câu hỏi:

Bạn luôn à, vậy là không dùng phương thức UART?

Trả lời:

Đúng rồi bạn, một số dòng chip đời mới (như ESP32-S3 hay ESP32-C3) đã có thể lược bỏ con chip UART rời trên mạch. Tuy nhiên, điểu này không có nghĩa là chúng bỏ hoàn toàn phương thức giao tiếp, mà là con chip ESP lúc này đã “thông minh” hơn.

Có hai lý do chính cho sự thay đổi này:

1. Tích hợp sẵn bộ chuyển đổi vào bên trong

Các dòng chip cũ, khi xử lý trung tâm không hiểu được tín hiệu USB. Nhưng các dòng mới, nhà sản xuất đã tích hợp sẵn một khối gọi là USB-Serial/JTAG Controller vào ngay bên trong con chip ESP.

Lúc này, bạn chỉ cần nối trực tiếp 2 dây dữ liệu của cáp USB (D+ và D-) vào thẳng 2 chân tương ứng trên con ESP.

Con chip ESP lúc này vừa đóng vai trò xử lý, vừa tự đóng vai trò là “người phiên dịch” luôn. Nó không cần con chip trung gian bên ngoài nữa.

2. Tiết kiệm và tối ưu

Việc bỏ chip UART rời (như CH340 hay CP2102) giúp:

Giảm chi phí: Bớt được tiền mua một con chip phụ.

Tiết kiệm diện tích: Bo mạch sẽ nhỏ gọn hơn rất nhiều.

Giảm tiêu thụ năng lượng: Không phải nuôi thêm một con chip chờ trên mạch.

Vậy nó còn dùng phương thức UART không?

Câu trả lời là: Vừa có, vừa không.

Về mặt vật lý: Không còn dùng các chân TX/RX để nạp code nữa mà dùng thẳng đường USB.

Về mặt phần mềm: Trên máy tính, khi bạn cắm chip vào, nó vẫn hiện lên một cái “Cổng COM ảo” (Virtual COM Port). Nghĩa là máy tính vẫn “tư tưởng” là đang nói chuyện qua UART, nhưng thực chất bên dưới là dữ liệu đang chạy qua đường USB trực tiếp.

Tóm lại:

Mạch cũ (ESP32 đời đầu): Cần chip UART rời để làm cầu nối.

Mạch mới (ESP32-S3, C3): Dùng USB trực tiếp (Native USB), chip tự xử lý tín hiệu USB luôn.

Đây chính là xu hướng hiện nay để làm cho các thiết bị điện tử ngày càng nhỏ và rẻ hơn.

...

Câu hỏi:

Đây chính phải là câu hỏi của tôi à, nhưng khác một chút chip esp32 là người nhận thư luôn.

Trả lời:

Chính xác luôn! Bạn ném bít vụn đi rất nhanh.

Đúng như bạn nói, 2 các dòng chip đời mới như E S P 32-S3, con chip này đóng hai vai trong cùng một thân xác: Vừa là “người nhận thư”, vừa là “người phiên dịch”.

Dưới đây là sự khác biệt giữa hai mô hình để bạn thấy rõ cái “khác một chút” đó:

1. Mô hình cũ (Phải có chip U A R T rời)

Giống như bạn mua gói thư cho một người nước ngoài nhưng người đó hoàn toàn không biết tiếng Anh.

Máy tính: Gửi thư bằng tiếng Anh (USB).

Chip U A R T rời: Đứng ngoài cửa, dịch thư từ tiếng Anh sang tiếng Việt (U A R T).

E S P 32 đời cũ: Nhận thư đã dịch bằng tiếng Việt và làm theo.

2. Mô hình mới (E S P 32 tích hợp USB nội bộ)

Giống như người nước ngoài đó đã tự đi học tiếng Anh rồi.

Máy tính: Gửi thư bằng tiếng Anh (USB).

E S P 32 (S3/C3): Có một bộ phận “học ngoại ngữ” nằm sẵn trong bụng (khởi USB Serial JTAG). Nó tự nhận luôn tín hiệu USB, tự hiểu và tự ghi vào Flash.

Tại sao đây là bước tiến lớn?

Việc con E S P 32 tự làm “người nhận thư” trực tiếp mang lại những lợi ích cực kỳ hay:

Nạp code “bít tít”: Với chip U A R T rời, nếu con chip đó cháy thì bạn không nạp được code nữa. Với loại tích hợp, đường truyền thẳng vào lõi chip nên rất khó hỏng.

Tốc độ: Vì không phải qua một con chip trung gian giới hạn tốc độ, việc nạp code và truy cập dữ liệu có thể nhanh hơn rất nhiều.

Giống lập thiết bị: Vì nó hiểu trực tiếp ngôn ngữ USB, bạn có thể lập trình để máy tính nhận diện con E S P 32 như là một cái bàn phím, một con chuột, hay một cái ổ đĩa USB thực thụ. Điều mà các dòng chip cũ phải qua chip U A R T rời rất khó thực hiện hoặc làm rất tốn kém.

Tóm lại: Nhận định của bạn hoàn toàn đúng cho các dòng E S P 32 hiện đại. Con chip trung gian đã bị “đuối việc” và con chip

chính đã kiêm luôn nhiệm vụ đó.

...

Câu hỏi:

Chip flash chip là trang vớ nên đâu có bootloader, đúng không bạn hay sai?

Trả lời:

Bạn tư duy rất chuẩn! Cách ví von “trang vớ” của bạn giúp làm sáng tỏ vấn đề này rất nhanh.

Chính xác là chip Flash không có Bootloader tự thân (nghĩa là nó không tự biết làm gì cả). Tuy nhiên, để hiểu trọn vẹn tại sao người ta vẫn hay nhắc đến “Bootloader trong Flash”, bạn có thể hình dung thế này:

1. Bản chất của chip Flash

Chip Flash đúng là một quyển vở trắng. Nếu bạn chỉ cắm điện cho riêng con chip Flash, nó sẽ đứng yên, không có hoạt động gì xảy ra vì nó không có bộ não (CPU) và không có “bản năng” (ROM) để tự chạy.

2. “Bootloader trong Flash” thực chất là gì?

Đây là điểm dễ gây nhầm lẫn nhất. Khi mọi người nói “Bootloader nằm trong Flash”, họ đang ám chỉ vị trí lưu trữ.

Người viết: Là chúng ta (lập trình viên).

Hành động: Chúng ta viết một đoạn code mới và nạp nó vào những trang đầu tiên của “quyển vở” Flash.

Người đọc: Con chip E S P 32.

Nói cách khác, Flash chip là nơi “cho thuê chỗ” để đặt chương trình Bootloader. Nếu không có con chip E S P 32 để đọc và thực thi những dòng chữ đó, thì đoạn Bootloader trong Flash cũng chỉ là những con số vô hồn.

3. Sự phối hợp giữa ROM và Flash

Để hệ thống hoạt động, cần một quy trình tiếp xúc:

Bước 1 (ROM): Khi có điện, E S P 32 chạy chương trình trong ROM (bản năng). Chương trình này báo chip: “Hãy đi kiểm tra chip Flash bên ngoài xem có ai viết gì ở đó không”.

Bước 2 (Flash): Nếu thấy có code ở đầu trang Flash, E S P 32 sẽ bắt đầu đọc và làm theo. Đoạn code đầu tiên nó đọc được chính là cái mà ta gọi là Bootloader tầng 2.

Tóm lại:

ROM: Có sẵn “bản năng” (Bootloader gốc) do nhà sản xuất nạp, không bao giờ mất, dùng để cứu hộ và nạp code ban đầu.

Flash: Chỉ là “trang vớ” chứa dữ liệu. Nó chứa Bootloader là

do chúng ta nạp vào để hỗ trợ việc quản lý các chương trình phức tạp hơn.

Bạn thấy đấy, chip Flash đúng là “thụ động” hoàn toàn. Mọi sự thông minh đều nằm ở con chip E S P 32.

...

Câu hỏi:

Tại sao người ta lại mất công nạp thêm một cái Bootloader vào Flash làm gì cho rắc rối, trong khi đã có cái trong ROM rồi nhỉ?

Trả lời:

Việc duy trì hai tầng Bootloader (một trong ROM và một trong Flash) nghe có vẻ công kềnh nhưng thực tế đây là một thiết kế cực kỳ thông minh để đảm bảo tính linh hoạt và an toàn cho con chip.

Dưới đây là những lý do chính tại sao người ta lại làm như vậy:

1. Khả năng cập nhật và tùy biến (Lý do quan trọng nhất)

ROM là bộ nhớ “chết”, nghĩa là một khi chip đã xuất xưởng, bạn không bao giờ thay đổi được nội dung bên trong đó. Nếu nhà sản xuất chỉ duy nhất một Bootloader trong ROM, bạn sẽ bị bó buộc mãi mãi vào cách hoạt động đó.

Trong khi đó, Bootloader trong Flash có thể cập nhật được. Khi công nghệ thay đổi hoặc phát triển thêm các tính năng mới như nạp code qua Wi-Fi (OTA – Over The Air), cập nhật bảo mật, hay mã hóa dữ liệu, chúng ta chỉ cần nạp lại đoạn Bootloader mới vào Flash là xong. Con chip sẽ trở nên thông minh hơn mà không cần thay đổi phần cứng.

2. Phân vùng và quản lý bộ nhớ phức tạp

Chương trình trong ROM rất nhỏ và đơn giản, nó chỉ biết “đọc và chạy”. Nhưng thực tế, một con E S P 32 có thể chứa cùng lúc nhiều chương trình khác nhau trên Flash (ví dụ: một bản chạy chính và một bản dự phòng để tự sửa lỗi).

Bootloader tầng 2 nằm trên Flash đóng vai trò như một “người quản lý thư viện”. Nó sẽ kiểm tra bảng phân vùng (Partition Table) để biết chương trình nào đang là bản mới nhất, kiểm tra xem code có bị lỗi không trước khi cho phép CPU thực thi. ROM không để “thông minh” và không để không gian để làm những việc quản lý phức tạp này.

3. Chế độ cứu hộ (Safety Net)

Hãy tưởng tượng nếu bạn lỡ tay nạp một chương trình lỗi vào

Flash làm con chip bị “treo” hoàn toàn. Nếu không có Bootloader trong ROM, con chip đó sẽ trở thành cục gạch vì bạn không còn cách nào để ra lệnh cho nó nhận code mới nữa.

Nhờ có Bootloader trong ROM luôn luôn tồn tại, bạn chỉ cần bấm giữ nút BOOT (hoặc kéo chân I00 xuống đất) khi cấp nguồn. Lúc này, E S P 32 sẽ đi qua đoạn Bootloader lưu trong Flash và quay về chạy “bản năng gốc” trong ROM. Nó sẽ mở lại cổng giao tiếp để bạn nạp lại code mới từ đầu. Đây chính là cái phao cứu sinh cho mọi lập trình viên.

Tóm lại:

Bootloader trong ROM là “Nền tảng an toàn”: Đảm bảo chip không bao giờ bị hỏng phần mềm hoàn toàn.

Bootloader trong Flash là “Trí khôn linh hoạt”: Cho phép chúng ta tùy biến cách chip khởi động, quản lý nhiều chương trình và cập nhật các tính năng hiện đại sau này.

...

Câu hỏi:

Vậy tại sao người ta lại bỏ chip flash mà lại dùng chip đơn esp32 cho nhiều dự án?

Trả lời:

Thực ra, trong kỹ thuật người ta không bỏ “bộ” bộ nhớ Flash, mà là họ chuyển từ việc dùng chip Flash rời sang dùng loại E S P 32 tích hợp Flash bên trong (gọi là dòng Internal Flash hoặc Embedded Flash).

Dưới đây là những lý do tại sao các dự án hiện đại thường ưu tiên loại “tất cả trong một” này:

1. Tiết kiệm không gian cực lớn

Trong các dự án như đồng hồ thông minh, thiết bị đeo tay (wearables) hoặc cảm biến mini, diện tích bo mạch rất quý giá. Việc tích hợp Flash vào bên trong chip chính giúp giảm kích thước mạch in (PCB) xuống đáng kể vì không còn cần chỗ cho con chip Flash rời và các đường dây nối lỏng ngoằn ngoèo giữa chúng.

2. Độ bền và độ tin cậy cao hơn

Khi dùng chip Flash rời, các chân đồng siêu nhỏ nối giữa E S P 32 và Flash có thể bị bong tróc, đứt gãy hoặc oxy hóa do môi trường (ẩm ướt, rung lắc). Khi Flash được đúc sẵn vào bên trong lõi chip, các kết nối này nằm trong lớp vỏ bảo vệ vĩnh viễn, giúp thiết bị hoạt động ổn định hơn trong các điều kiện khắc nghiệt.

3. Hiệu suất và tốc độ

Việc truy cập dữ liệu giữa chip xử lý và chip Flash bên ngoài bị giới hạn bởi tốc độ của các đường dây vật lý trên mạch (bus). Khi tích hợp bên trong, đường truy cập dữ liệu ngắn hơn và được tối ưu hóa về cấp độ vi mạch, giúp việc “load” chương trình và truy xuất dữ liệu diễn ra nhanh hơn và tiêu tốn ít năng lượng hơn.

4. Bền mật tốt hơn

Với chip Flash rời, một kẻ xấu có thể dùng kẹp hoặc máy đo tín hiệu để “nghe trộm” dữ liệu đang chạy trên các đường dây giữa hai con chip nhằm lấy cắp mã nguồn. Khi mọi thứ nằm chung trong một vỏ bọc Silicon duy nhất, việc can thiệp vào đường truy cập dữ liệu bên trong là gần như không thể đối với các thiết bị thông thường.

5. Đơn giản hóa sản xuất

Đối với các công ty sản xuất hàng nghìn thiết bị, việc chi phí quản lý một loại linh kiện (E S P 32) thay vì phải mua thêm chip Flash, quản lý kho bãi và hàn thêm một linh kiện nữa giúp giảm chi phí sản xuất và kiểm tra lỗi rất nhiều.

Tóm lại:

Việc dùng chip E S P 32 có sẵn Flash bên trong là xu hướng “All-in-One” (Tất cả trong một). Nó biến con chip từ một “bộ não” cần thêm “quyển vở” bên ngoài thành một thực thể hoàn chỉnh, có thể tự ghi nhớ và tự suy nghĩ ngay lập tức.

...

Câu hỏi:

Esp32 đơn nào đã tích hợp chip flash và cả chip uart?

Trả lời:

Nếu bạn muốn loại tối giản nhất, chỉ cần một con chip duy nhất mà “bụng” nó đã chứa sẵn cả Flash (để lưu code) và bộ nạp (để cắm thẳng vào máy tính) thì đây là những cái tên bạn cần tìm:

1. Dòng E S P 32-S3 (Mạnh mẽ và đa năng nhất)

Đây là “vua” của các loại tích hợp hiện nay.

Tích hợp Flash: Các mã như E S P 32-S3-FH4R2 đã đúc sẵn 4MB Flash bên trong lõi silicon.

Tích hợp bộ nạp: Nó có khe USB Serial/JTAG nội bộ. Bạn chỉ việc nối 2 dây dữ liệu từ cáp USB vào chân GPIO 19 và 20 của chip là máy tính nhận ngay cổng COM để nạp code. Không cần thêm bất kỳ con chip trung gian nào.

Điểm cộng: Nó cực mạnh, hỗ trợ cả AI (Xử lý giọng nói, hình ảnh) và có rất nhiều chân IO.

2. Dòng E S P 32-C3 (Giá rẻ, thay thế ESP8266)

Nếu dự án của bạn đơn giản hơn, cần tiết kiệm chi phí:

Tích hợp Flash: Hầu hết các biến thể C3 đều tích hợp sẵn 4MB Flash bên trong vỏ chip.

Tích hợp bộ nạp: Giống như S3, nó cũng có bộ điều khiển USB nội bộ. Bạn cắm thẳng vào máy tính qua 2 chân D+ và D- là xong.

Đặc điểm: Chỉ có 1 lõi (Single-core) và ít chân hơn S3, nhưng cực kỳ nhỏ gọn.

3. Dòng E S P 32-C6 (Dòng mới nhất)

Dòng này tương tự như C3 nhưng hiện đại hơn:

Tích hợp: Sẵn Flash và bộ nạp USB bên trong.

Điểm mới: Hỗ trợ Wi-Fi 6 và giao thức Matter (chuẩn nhà thông minh mới nhất).

Tóm tắt cách nhận diện:

Khi bạn mua chip lẻ để vẽ mạch, hãy để ý cái đuôi của tên chip:

Nếu thấy mã chip có chữ "H" (như FH4, FN8): Nghĩa là nó đã tích hợp sẵn Flash bên trong (H viết tắt của Has Flash).

Nếu dùng dòng S3, C3, C6: Nghĩa là nó đã tích hợp sẵn bộ nạp USB (không cần mua thêm chip nạp rời).

Lưu ý quan trọng: Dù chip đã tích hợp bộ nạp bên trong, nhưng khi bạn vẽ mạch, bạn vẫn nên đưa 2 chân USB D+ và D- ra một cổng cắm để có chỗ mà cắm cáp vào máy tính nhé!

4. Dưới đây là sự khác biệt để bạn chọn đúng con chip mình cần:

Lựa chọn 1: ESP32-S3-FH4R2 (Con chip mạnh mẽ)

Đây là một con chip đơn duy nhất. Bên trong lớp vỏ nhựa của nó đã có sẵn bộ não 2 nhân, 4MB Flash để lưu code và bộ nạp USB.

Bạn chọn con này khi dự án cần xử lý nhiều tác vụ nặng, cần nhiều chân cắm linh kiện hoặc làm vỏ camera, âm thanh.

Lựa chọn 2: ESP32-C3-FH4 (Con chip nhỏ gọn)

Đây cũng là một con chip đơn riêng biệt. Nó cũng tích hợp sẵn Flash và bộ nạp USB bên trong bụng nó luôn.

Bạn chọn con này khi muốn làm các thiết bị cực nhỏ, rẻ tiền như bóng đèn thông minh, cảm biến nhiệt độ Wi-Fi, nơi mà bạn không cần quá nhiều chân kết nối.

Điểm chung của cả hai con này:

Đều không cần chip nạp UART rời (như CH340).

Đều không cần chip Flash rời nằm bên cạnh.

Bạn chỉ cần cắm nguồn 3.3V và nối 2 dây dữ liệu USB vào đúng chân của nó là nó chạy.

Tóm lại:

Nếu bạn muốn làm một cái máy tính bảng mini hay thiết bị có màn hình phức tạp: Chọn S3.

Nếu bạn muốn làm một cái công tắc Wi-Fi nhỏ bằng đầu ngón tay: Chọn C3.

...

Câu hỏi:

Mô hình chip rời có bộ kit esp32 wroom?

Trả lời:

Chúc luôn! Bạn quan sát cực kỳ chính xác. Con WROOM bạn đang cầm chính là minh chứng rõ nhất cho mô hình “chip rời” mà chúng ta vừa bàn.

Đó mình làm rõ hơn cấu trúc của con đó để bạn đối chiếu với những gì bạn đang thấy trên mạch:

Cấu tạo thực tế của ESP32-WROOM bạn đang dùng

Con chip to nhất, vuông nhất ở giữa chính là ESP32 SoC. Bên trong nó có CPU và S RAM (bộ nhớ tạm thời, mất điện là mất dữ liệu). Nó giống như một “vi giám đốc” có trí nhớ ngắn hạn cực nhanh nhưng không có ngăn kéo để cất hồ sơ lâu dài.

Con chip nhỏ hơn 8 chân nằm ngay sát bên cạnh chính là Chip Flash rời. Đây mới là nơi chứa code bạn nạp từ Arduino IDE vào. Khi bạn cấp điện, con chip xử lý sẽ “gọi” dữ liệu từ con chip Flash 8 chân này qua các đường mạch đồng nhỏ xíu (chuẩn SPI) để bắt đầu làm việc.

Tại sao bạn lại thấy nó nằm ngoài?

Vì dòng WROOM này được thiết kế theo kiểu module lắp ghép. Việc để Flash nằm ngoài giúp nhà sản xuất dễ dàng thay đổi dung lượng (bạn có thể thấy loại 4MB, 8MB hay 16MB cùng một kiểu dáng) mà không phải sản xuất lại con chip xử lý chính.

Sự khác biệt so với loại “tích hợp” mình nói lúc nãy

Với các dòng đời mới như E S P 32-S3-FH4 hay E S P 32-C3-FH4 mà mình nhắc tới, nếu bạn có cây nẹp sắt ra hay nhìn dưới kính hiển vi, bạn cũng sẽ không thấy con chip 8 chân nằm kề bên nữa. Nó đã bị “nuốt” vào trong lớp vỏ đen của con chip xử lý chính luôn rồi.

Vậy là con bạn đang dùng vốn thuộc thể hệ “truyền thống”:

Chip xử lý riêng (chứa CPU, S RAM, ROM).

Chip Flash riêng (nằm kề bên).

Chip U A R T rời (nằm trên bo mạch DevKit, phía gần cổng USB).

...

Câu hỏi:

Cách đọc mã trên con chip flash?

Trả lời:

Cách đọc mã trên con chip Flash 8 chân rất thú vị vì nó giúp bạn biết chính xác “quyển vở” của mình dày bao nhiêu trang.

Dưới đây là cách giải mã các ký hiệu phổ biến nhất mà bạn thường thấy trên các module ESP32:

Cấu trúc mã chip Flash phổ biến

Hầu hết các chip Flash rời trên ESP32 đến từ các hãng như Winbond, GigaDevice hoặc AI-Thinker. Mã của chúng thường có dạng: 25QXX.

Trong đó, con số XX sau chữ Q chính là chìa khóa để biết dung lượng theo đơn vị Megabit (Mb). Để đổi sang Megabyte (MB) mà chúng ta hay dùng, bạn chỉ cần lấy con số đó chia cho 8.

Danh sách các mã thường gặp

25Q80: Có dung lượng 8 Megabit, tương đương 1 Megabyte (MB).

25Q16: Có dung lượng 16 Megabit, tương đương 2 Megabyte (MB).

25Q32: Có dung lượng 32 Megabit, tương đương 4 Megabyte (MB).

Đây là loại phổ biến nhất trên các dòng WROOM.

25Q64: Có dung lượng 64 Megabit, tương đương 8 Megabyte (MB).

25Q128: Có dung lượng 128 Megabit, tương đương 16 Megabyte (MB).

Cách tính nhanh trong đầu

Cứ mỗi số 8 đơn vị Megabit sẽ tạo thành 1 đơn vị Megabyte.

Ví dụ: Nếu bạn thấy chip ghi 25Q32, bạn lấy $32 / 8 = 4$. Vậy mạch của bạn có 4MB Flash.

Các ký tự phụ khác

Phía sau con số dung lượng thường có thêm các chữ cái như BSIG, FVSIG... Những chữ này chỉ yêu nói về tốc độ bus, điện áp hoạt động hoặc kiểu đóng gói chân cắm, không ảnh hưởng đến dung lượng lưu trữ code của bạn.